

# Congetture Fondamentali della Crittografia Moderna

La crittografia moderna si fonda su un paradosso fondamentale: i sistemi che proteggono le comunicazioni globali, le transazioni finanziarie e i segreti di stato poggiano su assunzioni matematiche non dimostrate. A differenza di altri ambiti della matematica applicata, dove teoremi consolidati garantiscono la correttezza degli algoritmi, la sicurezza crittografica dipende dalla presunta difficoltà computazionale di problemi per i quali non esiste alcuna dimostrazione rigorosa di intrattabilità. Questa situazione non rappresenta una debolezza metodologica della crittografia, ma piuttosto riflette la profondità straordinaria dei problemi computazionali che la disciplina utilizza come fondamento. La distinzione cruciale tra dimostrazione e congettura diventa particolarmente significativa quando consideriamo l'infrastruttura crittografica globale. Ogni transazione con carta di credito, ogni connessione HTTPS, ogni firma digitale su documenti legali si basa sulla congettura che certi problemi computazionali richiedano tempo superpolinomiale per essere risolti. Non esiste alcun teorema che garantisca questa proprietà. Esistono solo decenni di tentativi falliti di trovare algoritmi efficienti, accompagnati da argomentazioni euristiche e riduzioni tra problemi che suggeriscono, ma non dimostrano, la durezza computazionale. In questa sede accendiamo alle congetture matematiche fondamentali su cui poggia l'intera architettura della crittografia contemporanea, analizzando con rigore tecnico la loro formulazione, le interconnessioni tra di esse, i risultati parziali ottenuti e le implicazioni pratiche per i sistemi crittografici reali. Procederemo dall'esame delle congetture classiche che hanno dominato la crittografia per decenni fino alle assunzioni più recenti che sottendono gli schemi post-quantistici attualmente in fase di standardizzazione.

## Parte I: Il Problema del Logaritmo Discreto e le sue Varianti

### Formulazione Matematica e Contesto Algebrico

Il problema del logaritmo discreto rappresenta uno dei pilastri fondamentali della crittografia a chiave pubblica. Per comprenderne appieno la natura e le implicazioni, dobbiamo iniziare con una formulazione algebrica rigorosa che ne catturi la struttura essenziale.

Sia  $G$  un gruppo finito ciclico di ordine  $n$ , scritto moltiplicativamente, e sia  $g$  un generatore di  $G$ . Questo significa che ogni elemento  $h \in G$  può essere espresso come  $h = g^x$  per qualche intero  $x$  con  $0 \leq x < n$ . L'operazione di esponenziazione  $g^x$  può essere calcolata efficientemente mediante l'algoritmo di esponenziazione rapida, che richiede  $O(\log x)$  operazioni di gruppo. Questo algoritmo sfrutta la rappresentazione binaria dell'esponente: se  $x = \sum_{i=0}^k b_i \cdot 2^i$  con  $b_i \in \{0,1\}$ , allora  $g^x = \prod_{i:b_i=1} g^{2^i}$ , e i valori  $g^{2^i}$  possono essere precalcolati mediante ripetuti quadrati.

Il problema del logaritmo discreto chiede di invertire questa operazione: dato il generatore  $g$  e un elemento  $h \in G$ , trovare l'intero  $x$  tale che  $g^x = h$ . Questo intero  $x$  è chiamato il

logaritmo discreto di  $h$  in base  $g$ , spesso denotato come  $\log_g(h)$  o  $\text{dlog}_g(h)$ . La congettura fondamentale asserisce che, per gruppi opportunamente scelti e ordini sufficientemente grandi, non esiste alcun algoritmo che risolva questo problema in tempo polinomiale nella lunghezza della rappresentazione di  $n$ .

Formuliamo questa congettura con maggiore precisione. Sia  $\ell(n)$  la lunghezza in bit necessaria per rappresentare elementi di un gruppo di ordine  $n$ , tipicamente  $\ell(n) = \lceil \log_2(n) \rceil$ . Un algoritmo risolve il problema del logaritmo discreto in tempo polinomiale se il suo tempo di esecuzione è limitato da un polinomio in  $\ell(n)$ . La congettura del logaritmo discreto, nella sua forma più generale, afferma che per certe famiglie di gruppi parametrizzate dalla dimensione, non esiste alcun algoritmo deterministico o probabilistico che risolva il problema in tempo polinomiale.

La difficoltà del logaritmo discreto dipende criticamente dalla scelta del gruppo  $G$ . Per il gruppo additivo  $\mathbb{Z}_n$ , il problema è banale: dato  $g$  e  $h$ , il logaritmo discreto è semplicemente  $x = h/g \bmod n$ , calcolabile mediante l'algoritmo euclideo esteso. Questa osservazione evidenzia che la durezza del logaritmo discreto non è una proprietà intrinseca dei gruppi finiti, ma emerge solo per specifiche classi di gruppi con struttura algebrica appropriata.

I due gruppi più rilevanti per applicazioni crittografiche sono il gruppo moltiplicativo  $F_p^*$  di un campo finito primo e i gruppi di punti di curve ellittiche su campi finiti. Per  $F_p^*$ , il problema del logaritmo discreto è formulato come segue: dato un primo  $p$ , un generatore  $g$  dell'ordine del sottogruppo ciclico (tipicamente un sottogruppo di ordine  $q$  dove  $q$  è un divisore primo grande di  $p-1$ ), e un elemento  $h \in F_p^*$ , trovare  $x$  tale che  $g^x \equiv h \pmod{p}$ . Per curve ellittiche, il problema è analogo ma formulato in notazione additiva: data una curva ellittica  $E$  definita su un campo finito  $F_q$ , un punto base  $P$  di ordine  $n$ , e un punto  $Q \in E(F_q)$ , trovare l'intero  $x$  tale che  $[x]P = Q$ , dove  $[x]P$  denota la moltiplicazione scalare del punto  $P$  per  $x$ .

## La Famiglia di Problemi Diffie-Hellman

Il protocollo di scambio di chiavi Diffie-Hellman, che ha rivoluzionato la crittografia quando fu pubblicato nel 1976, introduce una variante del problema del logaritmo discreto che risulta essere più debole in un senso tecnico preciso, ma comunque sufficiente per garantire la sicurezza del protocollo.

Il problema computazionale di Diffie-Hellman (CDH) è formulato come segue: dato un gruppo ciclico  $G$  di ordine  $n$  con generatore  $g$ , e dati elementi  $g^a$  e  $g^b$  per interi  $a, b$  scelti uniformemente a caso da  $\{0, 1, \dots, n-1\}$ , calcolare  $g^{ab}$ . Notiamo che questo problema chiede solo di calcolare un elemento specifico del gruppo, non di recuperare gli esponenti segreti  $a$  o  $b$ . È immediato osservare che se possiamo risolvere il problema del logaritmo discreto, possiamo risolvere CDH: basta calcolare  $a = \log_g(g^a)$ , e poi calcolare  $(g^b)^a = g^{ab}$ . Quindi CDH non può essere più difficile del logaritmo discreto. La questione aperta è se CDH sia strettamente più facile del logaritmo discreto, cioè se esista un algoritmo che risolva CDH senza risolvere il logaritmo discreto.

Per oltre quattro decenni, non è stato trovato alcun algoritmo di questo tipo per i gruppi standard utilizzati in crittografia. Questa evidenza empirica supporta la congettura che CDH e il logaritmo discreto abbiano essenzialmente la stessa difficoltà computazionale, almeno

per i gruppi comunemente impiegati. Tuttavia, dal punto di vista della teoria della complessità, rimane concepibile che CDH sia strettamente più facile, e questa possibilità rappresenta un rischio teorico per la sicurezza dei protocolli basati su Diffie-Hellman.

Una variante ancora più debole è il problema decisionale di Diffie-Hellman (DDH). Questo problema chiede di distinguere tra due distribuzioni: la prima è costituita da quadruple  $(g, g^a, g^b, g^{ab})$  dove  $a, b$  sono scelti uniformemente a caso; la seconda consiste in quadruple  $(g, g^a, g^b, g^c)$  dove  $a, b, c$  sono tutti indipendenti e uniformemente casuali. Un algoritmo risolve DDH se, data una quadrupla scelta da una delle due distribuzioni, può determinare con probabilità significativamente maggiore di  $1/2$  da quale distribuzione proviene.

La struttura di queste riduzioni può essere rappresentata mediante la catena di implicazioni: se il logaritmo discreto è facile, allora CDH è facile, e se CDH è facile, allora DDH è facile. Formalmente, scriviamo  $DLog \leq CDH \leq DDH$ , dove  $\leq$  denota riduzione polinomiale. Queste riduzioni sono tutte nella direzione "facile": mostrare che la difficoltà fluisce verso il basso. Le questioni aperte riguardano le riduzioni inverse: è DDH equivalente a CDH? È CDH equivalente a DLog? Per gruppi particolari, conosciamo separazioni. Ad esempio, per gruppi in cui esiste un accoppiamento bilineare (pairing)  $e : G \times G \rightarrow G_T$ , il problema DDH diventa facile: per verificare se  $(g, g^a, g^b, g^c)$  è una quadrupla Diffie-Hellman valida, basta controllare se  $e(g^a, g^b) = e(g, g^c)$ . Ma per gruppi generali, e in particolare per gruppi di curve ellittiche senza pairing efficienti, DDH rimane congetturalmente difficile.

## Algoritmi per il Logaritmo Discreto: Limiti Teorici e Pratici

Comprendere gli algoritmi noti per risolvere il logaritmo discreto è essenziale per valutare la solidità delle congetture di durezza e per scegliere parametri crittografici sicuri. Gli algoritmi esistenti si dividono in due categorie: algoritmi generici che funzionano per qualsiasi gruppo rappresentato come scatola nera, e algoritmi specifici che sfruttano la struttura particolare di certi gruppi.

Gli algoritmi generici operano senza assumere alcuna conoscenza della rappresentazione interna del gruppo. Possono solo eseguire operazioni di gruppo (moltiplicazione e inversione) e test di uguaglianza. Il primo algoritmo non banale di questa classe è l'algoritmo baby-step giant-step di Shanks. L'idea è di scrivere il logaritmo discreto  $x$  nella forma  $x = i \cdot m + j$  dove  $m = \lceil \sqrt{n} \rceil$  e  $0 \leq i, j < m$ . Allora  $h = g^x = g^{(im+j)}$  implica  $h \cdot g^{(-im)} = g^j$ . L'algoritmo precalcola tutti i valori  $g^j$  per  $j = 0, 1, \dots, m-1$  e li memorizza in una tabella hash. Poi, per ogni  $i = 0, 1, \dots, m-1$ , calcola  $h \cdot g^{(-im)}$  e verifica se questo valore appare nella tabella. Quando trova una corrispondenza, ha trovato  $i$  e  $j$ , quindi  $x = im + j$ .

L'algoritmo baby-step giant-step richiede  $O(\sqrt{n})$  operazioni di gruppo e  $O(\sqrt{n})$  spazio di memoria. Per un gruppo di ordine  $n \approx 2^k$ , questo significa  $O(2^{k/2})$  tempo e spazio. Questo rappresenta un miglioramento esponenziale rispetto alla ricerca esaustiva, che richiederebbe  $O(n) = O(2^k)$  tempo, ma rimane ancora esponenziale nella dimensione del logaritmo di  $n$ .

L'algoritmo rho di Pollard migliora l'algoritmo di Shanks eliminando il requisito di memoria esponenziale, mantenendo però la stessa complessità temporale  $O(\sqrt{n})$ . L'algoritmo definisce una funzione pseudo-casuale  $f : G \rightarrow G$  che mappa elementi del gruppo in altri

elementi, e costruisce una sequenza  $x_0, x_1, x_2, \dots$  dove  $x_{i+1} = f(x_i)$ . Questa sequenza, essendo deterministica e definita su un insieme finito, deve eventualmente entrare in un ciclo. L'algoritmo cerca una collisione  $x_i = x_j$  con  $i < j$ , usando la tecnica di Floyd del "tortoise and hare" per rilevare il ciclo con spazio  $O(1)$ . La funzione  $f$  è costruita in modo che ogni elemento  $x_i$  possa essere espresso come  $x_i = g^{a_i} \cdot h^{b_i}$  per interi  $a_i, b_i$  che l'algoritmo mantiene esplicitamente. Quando trova una collisione  $x_i = x_j$ , ottiene  $g^{a_i} \cdot h^{b_i} = g^{a_j} \cdot h^{b_j}$ , e sapendo che  $h = g^x$ , deriva  $(a_i - a_j) + x(b_i - b_j) \equiv 0 \pmod{n}$ , da cui  $x = (a_j - a_i)/(b_i - b_j) \pmod{n}$ , calcolabile se  $\gcd(b_i - b_j, n) = 1$ .

Un risultato teorico fondamentale di Shoup stabilisce un limite inferiore per algoritmi generici. Shoup dimostrò che qualsiasi algoritmo generico per il logaritmo discreto deve eseguire almeno  $\Omega(\sqrt{n})$  operazioni di gruppo. Questo risultato implica che algoritmi come baby-step giant-step e rho di Pollard sono asintoticamente ottimali tra tutti gli algoritmi generici. La dimostrazione di Shoup utilizza tecniche di teoria dell'informazione: mostra che dopo meno di  $c\sqrt{n}$  operazioni per una costante appropriata  $c$ , la distribuzione delle risposte osservate dall'algoritmo non può distinguere il valore corretto di  $x$  da altri valori candidati con probabilità sufficientemente alta.

Il risultato di Shoup ha implicazioni profonde per la crittografia. Garantisce che per gruppi senza struttura sfruttabile, la sicurezza del logaritmo discreto con parametri di dimensione  $k$  bit richiede almeno  $2^{(k/2)}$  operazioni per essere violata. Questo è il motivo per cui, nella pratica, si usano gruppi di ordine almeno  $2^{256}$  per garantire sicurezza a 128 bit contro algoritmi generici.

Tuttavia, per gruppi con struttura algebrica particolare, esistono algoritmi sub-esponenziali che violano il limite di Shoup sfruttando rappresentazioni specifiche. Il più importante è l'algoritmo index calculus per il gruppo moltiplicativo  $F_p^*$ . Questo algoritmo, nelle sue varie incarnazioni (index calculus classico, function field sieve, number field sieve), raggiunge complessità sub-esponenziale della forma  $L_p[\alpha, c] = \exp((c + o(1)) \cdot (\log p)^\alpha \cdot (\log \log p)^{(1-\alpha)})$  per certi valori di  $\alpha$  e  $c$ . Per il number field sieve applicato a  $F_p^*$ , la complessità è  $L_p[1/3, (64/9)^{1/3}] \approx \exp(1.923 \cdot (\log p)^{1/3} \cdot (\log \log p)^{2/3})$ .

L'idea fondamentale di index calculus è di lavorare con un insieme di "primi piccoli"  $B = \{p_1, p_2, \dots, p_k\}$  chiamato base di fattori. L'algoritmo cerca relazioni della forma  $g^r = \prod_{i=1}^k p_i^{e_i}$  per interi casuali  $r$ . Prendendo logaritmi, otteniamo  $r \equiv \sum_{i=1}^k e_i \cdot \log_g(p_i) \pmod{p-1}$ . Dopo aver raccolto sufficienti relazioni lineari (più di  $k$ ), il sistema può essere risolto per i logaritmi  $\log_g(p_i)$  degli elementi della base di fattori. Una volta noti questi valori, per calcolare  $\log_g(h)$  per un  $h$  arbitrario, si cerca un intero  $s$  tale che  $h \cdot g^s$  si fattorizzi completamente sulla base  $B$ :  $h \cdot g^s = \prod p_i^{f_i}$ . Allora  $\log_g(h) = \sum f_i \cdot \log_g(p_i) - s$ .

La fase più costosa è trovare relazioni smooth, cioè trovare esponenti  $r$  tali che  $g^r$  si fattorizzi sulla base  $B$ . La probabilità che un elemento casuale di  $F_p$  sia  $B$ -smooth (cioè che tutti i suoi fattori primi siano in  $B$ ) è governata dalla funzione di Dickman, e questa probabilità decresce rapidamente all'aumentare di  $p$ . Bilanciare la dimensione della base di fattori con la probabilità di smoothness porta alla complessità sub-esponenziale citata.

È cruciale notare che index calculus non si applica facilmente ai gruppi di curve ellittiche. La ragione è che non esiste un analogo naturale della fattorizzazione per punti su curve

ellittiche. Un punto  $P$  su una curva ellittica  $E$  non può essere "fattorizzato" in "punti primi" in modo significativo. Vari tentativi di adattare index calculus alle curve ellittiche sono stati proposti, come l'algoritmo di Semaev-Gaudry-Diem, ma questi funzionano solo per curve con proprietà molto speciali e richiedono tempo esponenziale per curve generali. Questo è uno dei motivi principali per cui le curve ellittiche offrono sicurezza superiore rispetto a  $F_p^*$  per dimensioni di chiave comparabili: l'assenza di algoritmi sub-esponenziali noti per il logaritmo discreto su curve ellittiche generali.

## Implicazioni per Protocolli Crittografici Reali

La congettura di durezza del logaritmo discreto sostiene una vasta gamma di protocolli crittografici fondamentali. Lo scambio di chiavi Diffie-Hellman permette a due parti, Alice e Bob, di stabilire una chiave segreta condivisa su un canale pubblico. Il protocollo procede come segue: Alice sceglie un esponente segreto  $a$  uniformemente a caso da  $\{1, \dots, n-1\}$  e invia  $g^a$  a Bob. Bob sceglie similmente un esponente segreto  $b$  e invia  $g^b$  ad Alice. Alice calcola  $(g^b)^a = g^{(ab)}$ , e Bob calcola  $(g^a)^b = g^{(ab)}$ . Entrambi ottengono lo stesso valore  $g^{(ab)}$ , che diventa la chiave segreta condivisa. Un attaccante che osserva lo scambio vede solo  $g$ ,  $g^a$ ,  $g^b$ , ma non  $a$ ,  $b$  o  $g^{(ab)}$ . La sicurezza del protocollo si basa sulla congettura CDH: calcolare  $g^{(ab)}$  da  $g^a$  e  $g^b$  deve essere computazionalmente difficile.

Il sistema di cifratura ElGamal estende questa idea a uno schema di cifratura a chiave pubblica. La chiave pubblica è una coppia  $(g, h)$  dove  $h = g^x$  per un esponente segreto  $x$  che costituisce la chiave privata. Per cifrare un messaggio  $m \in G$ , si sceglie un esponente casuale  $y$  e si calcola il testo cifrato  $(c_1, c_2) = (g^y, m \cdot h^y)$ . Per decifrare, si calcola  $m = c_2 / (c_1^x) = c_2 / (g^{xy}) = (m \cdot h^y) / (g^{xy}) = m \cdot g^{xy} / g^{xy} = m$ . La sicurezza semantica di ElGamal si basa sulla congettura DDH: la coppia  $(g^y, h^y)$  deve essere indistinguibile da una coppia casuale  $(g^y, g^z)$  dove  $z$  è indipendente da  $y$ .

Il Digital Signature Algorithm (DSA) e il suo omologo per curve ellittiche (ECDSA) forniscono schemi di firma digitale basati sul logaritmo discreto. In DSA, i parametri pubblici includono un primo grande  $p$ , un sottogruppo di ordine primo  $q$  che divide  $p-1$ , e un generatore  $g$  di questo sottogruppo. La chiave privata è un intero  $x$  scelto da  $\{1, \dots, q-1\}$ , e la chiave pubblica è  $y = g^x \bmod p$ . Per firmare un messaggio  $m$ , si sceglie un nonce casuale  $k$  da  $\{1, \dots, q-1\}$ , si calcola  $r = (g^k \bmod p) \bmod q$  e  $s = k^{-1} \cdot (H(m) + x \cdot r) \bmod q$ , dove  $H$  è una funzione hash crittografica. La firma è la coppia  $(r, s)$ . Per verificare, si calcola  $w = s^{-1} \bmod q$ ,  $u_1 = H(m) \cdot w \bmod q$ ,  $u_2 = r \cdot w \bmod q$ , e  $v = (g^{u_1} \cdot y^{u_2} \bmod p) \bmod q$ . La firma è valida se e solo se  $v = r$ .

La sicurezza di DSA richiede che sia difficile, dato  $(g, y, r, s, m)$ , calcolare la chiave privata  $x$  o forgiare una firma valida per un messaggio diverso. Questo si riduce essenzialmente al problema del logaritmo discreto: se un attaccante può calcolare  $x$  da  $y = g^x$ , può firmare messaggi arbitrari. Inoltre, la scelta del nonce  $k$  deve essere veramente casuale e mai riutilizzata. Se lo stesso  $k$  viene usato per firmare due messaggi diversi  $m_1$  e  $m_2$ , un attaccante può recuperare la chiave privata  $x$ . Infatti, dalle due firme  $(r, s_1)$  e  $(r, s_2)$  con lo stesso  $r$ , si ha  $s_1 - s_2 = k^{-1} \cdot (H(m_1) - H(m_2)) \bmod q$ , da cui  $k = (H(m_1) - H(m_2)) / (s_1 - s_2) \bmod q$ . Poi, da  $s_1 = k^{-1} \cdot (H(m_1) + x \cdot r)$ , si ricava  $x = (k \cdot s_1 - H(m_1)) / r \bmod q$ . Questo attacco è stato sfruttato in incidenti reali, come la compromissione della

chiave privata utilizzata per firmare aggiornamenti della PlayStation 3, dove Sony aveva usato un nonce statico.

## Varianti e Generalizzazioni del Problema

Il logaritmo discreto ha numerose varianti che appaiono in contesti crittografici diversi. Il problema del logaritmo discreto rappresentazionale chiede, dati elementi  $g_1, \dots, g_n, h$  di un gruppo  $G$ , di trovare esponenti  $x_1, \dots, x_n$  tali che  $h = g_1^{x_1} \cdot \dots \cdot g_n^{x_n}$ . Questo generalizza il logaritmo discreto standard, che corrisponde al caso  $n = 1$ . Il problema è rilevante per sistemi crittografici basati su rappresentazioni di gruppo, come certi schemi di identificazione e firme digitali.

Un'altra generalizzazione importante è il problema del logaritmo discreto vettoriale, che emerge in sistemi crittografici su reticoli ideali e in schemi basati su isogenie. In questi contesti, il "gruppo" sottostante non è abeliano, e il problema diventa significativamente più complesso. Per certi gruppi non abeliani, il problema del logaritmo discreto può essere più facile che per gruppi abeliani di dimensione comparabile, ma per altri gruppi non abeliani attentamente scelti, il problema sembra rimanere difficile.

Il problema del logaritmo discreto in gruppi con struttura aggiuntiva ha ricevuto particolare attenzione. Per gruppi con accoppiamenti bilineari efficienti, come certi gruppi di punti su curve ellittiche supersingolari o curve con embedding degree piccolo, il problema può essere trasferito dal gruppo originale a un campo finito più grande mediante l'accoppiamento. Questo attacco, chiamato riduzione MOV dal nome dei suoi scopritori Menezes, Okamoto e Vanstone, funziona come segue. Sia  $e : G \times G \rightarrow G_T$  un accoppiamento bilineare, e sia  $h = g^x$  il problema da risolvere. Allora  $e(h, g) = e(g^x, g) = e(g, g)^x$ . Quindi il logaritmo discreto di  $h$  in base  $g$  nel gruppo  $G$  si riduce al logaritmo discreto di  $e(h, g)$  in base  $e(g, g)$  nel gruppo  $G_T$ . Se  $G_T$  è isomorfo al gruppo moltiplicativo di un campo finito, possiamo applicare index calculus, potenzialmente rompendo il sistema.

Questa osservazione ha portato alla categorizzazione di curve ellittiche in "pairing-friendly" e "non pairing-friendly". Le curve pairing-friendly sono intenzionalmente scelte per avere accoppiamenti efficienti e sono utilizzate in crittografia basata su pairing, dove la bilinearità dell'accoppiamento viene sfruttata per costruire protocolli avanzati come Identity-Based Encryption e firme corte. Le curve non pairing-friendly, che costituiscono la stragrande maggioranza delle curve ellittiche, non hanno accoppiamenti efficientemente calcolabili e sono usate per crittografia standard dove vogliamo che il logaritmo discreto rimanga difficile.

## Parte II: La Durezza della Fattorizzazione e l'Infrastruttura RSA

### Il Problema della Fattorizzazione: Formulazione e Contesto

Il problema della fattorizzazione è uno dei problemi computazionali più antichi e meglio studiati in matematica, con una storia che risale almeno a Fermat nel XVII secolo. Nella sua formulazione moderna per la crittografia, il problema è il seguente: dato un intero positivo  $n$  che è il prodotto di due primi distinti  $p$  e  $q$ , cioè  $n = p \cdot q$ , trovare  $p$  e  $q$ . La durezza di questo

problema costituisce la base di sicurezza per il sistema crittografico RSA, che rimane uno degli schemi a chiave pubblica più ampiamente utilizzati nonostante abbia quasi cinquant'anni di età.

La congettura di durezza della fattorizzazione afferma che non esiste alcun algoritmo che fattorizzi interi composti in tempo polinomiale nella lunghezza della loro rappresentazione binaria. Più precisamente, se  $n$  ha lunghezza  $\ell$  bit, cioè  $2^{\ell-1} \leq n < 2^\ell$ , allora qualsiasi algoritmo che fattorizzi  $n$  richiede tempo superpolinomiale in  $\ell$ . La formulazione esatta della congettura dipende dal modello computazionale considerato: per computer classici deterministici, per computer classici probabilistici, o per computer quantistici.

È importante distinguere tra diverse varianti del problema di fattorizzazione che appaiono nella letteratura crittografica. Il problema decisionale chiede semplicemente se un dato intero  $n$  è primo o composto. Questo problema è stato risolto in tempo polinomiale deterministico dagli algoritmi AKS (Agrawal-Kayal-Saxena) nel 2002, quindi sappiamo che il test di primalità è in P. Il problema di ricerca chiede di trovare un fattore non banale di  $n$  se  $n$  è composto. Il problema della fattorizzazione completa chiede di trovare la fattorizzazione completa di  $n$  in fattori primi. Per  $n = p \cdot q$  con  $p$  e  $q$  primi, il problema di ricerca e la fattorizzazione completa coincidono essenzialmente: trovare un fattore  $p$  permette immediatamente di calcolare l'altro fattore  $q = n/p$ .

Il teorema fondamentale dell'aritmetica garantisce che ogni intero  $n > 1$  ha una fattorizzazione unica in primi:  $n = p_1^{a_1} \cdot p_2^{a_2} \cdot \dots \cdot p_k^{a_k}$  dove  $p_1 < p_2 < \dots < p_k$  sono primi e  $a_i \geq 1$ . La fattorizzazione è unica a meno dell'ordine dei fattori. Computare questa fattorizzazione, però, è computazionalmente difficile per interi grandi anche quando sappiamo che esiste ed è unica.

## Algoritmi Classici per la Fattorizzazione

Gli algoritmi di fattorizzazione si dividono in due categorie: metodi di scopo generale che funzionano per qualsiasi intero  $n$  indipendentemente dalla struttura dei suoi fattori, e metodi speciali che sono efficienti solo quando  $n$  ha fattori con proprietà particolari. I metodi di scopo generale sono quelli rilevanti per la sicurezza di RSA, poiché nella pratica crittografica i primi  $p$  e  $q$  sono scelti per non avere proprietà speciali sfruttabili.

Il metodo più semplice è la divisione per tentativi: provare a dividere  $n$  per ogni intero  $k = 2, 3, 4, \dots, \sqrt{n}$  fino a trovare un divisore. Se  $n = p \cdot q$  con  $p \leq q$ , allora  $p \leq \sqrt{n}$ , quindi questo metodo trova sicuramente  $p$ . La complessità è  $O(\sqrt{n})$ , che per  $n \approx 2^\ell$  è  $O(2^{\ell/2})$ , esponenziale in  $\ell$ . Per un modulo RSA a 2048 bit, questo richiederebbe circa  $2^{1024}$  operazioni, completamente infattibile.

Il metodo rho di Pollard per la fattorizzazione è concettualmente simile al suo omologo per il logaritmo discreto. L'algoritmo costruisce una sequenza pseudo-casuale  $x_0, x_1, x_2, \dots$  in  $\mathbb{Z}_n$  mediante una funzione  $f(x) = x^2 + a \pmod n$  per qualche costante  $a$ . Se  $n$  ha un fattore primo  $p$ , la sequenza  $x_i \pmod p$  diventa eventualmente periodica con periodo molto più piccolo del periodo della sequenza in  $\mathbb{Z}_n$ . Quando la sequenza mod  $p$  entra in un ciclo, avremo  $x_i \equiv x_j \pmod p$  per qualche  $i < j$ , ma tipicamente  $x_i \not\equiv x_j \pmod n$ . Quindi  $\gcd(x_i - x_j, n)$  sarà un multiplo di  $p$  ma non di  $n$ , rivelando un fattore non banale. L'algoritmo calcola  $\gcd(x_i - x_j, n)$  per coppie  $i, j$  usando la tecnica di Floyd per rilevare cicli. La complessità

attesa è  $O(n^{1/4}) = O(2^{l/4})$ , ancora esponenziale ma migliore della divisione per tentativi.

Il metodo  $p-1$  di Pollard sfrutta la struttura del gruppo moltiplicativo  $Z_p^*$ . Se  $p$  è un fattore primo di  $n$  e se  $p-1$  è  $B$ -smooth per qualche bound  $B$  non troppo grande, il metodo può trovare  $p$  efficientemente. L'algoritmo sceglie una base casuale  $a \in Z_n$  e calcola  $b = a^{(k!)} \bmod n$  dove  $k$  è scelto tale che tutti i primi minori di  $B$  dividano  $k!$ . Se  $p-1$  è  $B$ -smooth, allora  $p-1$  divide  $k!$ , quindi per il teorema di Fermat  $a^{(p-1)} \equiv 1 \pmod{p}$ , e quindi  $a^{(k!)} \equiv 1 \pmod{p}$ . Ma  $a^{(k!)} \bmod n$  sarà tipicamente diverso da 1 in  $Z_n$ . Quindi  $\gcd(a^{(k!)} - 1, n)$  rivelerà il fattore  $p$ . La complessità dipende da  $B$ : per  $B$  piccolo, l'algoritmo è veloce ma funziona solo per  $n$  con fattori smooth. Nella pratica crittografica,  $p$  e  $q$  sono scelti tali che  $p-1$  e  $q-1$  abbiano entrambi almeno un fattore primo grande, rendendo questo metodo inefficace.

Il metodo delle curve ellittiche di Lenstra (ECM) generalizza l'idea del metodo  $p-1$  lavorando con gruppi di punti su curve ellittiche casuali modulo  $n$ . Per ogni curva ellittica  $E$  definita su  $Z_n$ , la riduzione modulo un fattore primo  $p$  di  $n$  produce una curva ellittica su  $F_p$  il cui ordine del gruppo di punti varia con la curva scelta. Provando molte curve, si spera di trovarne una il cui ordine modulo  $p$  sia smooth, permettendo la fattorizzazione. ECM ha complessità euristica  $\exp((\sqrt{2} + o(1)) \cdot \sqrt{\log p \cdot \log \log p})$  per trovare un fattore  $p$ , quindi la complessità dipende dalla dimensione del fattore più piccolo piuttosto che dalla dimensione di  $n$ . Questo rende ECM particolarmente efficace per trovare fattori piccoli o medi di numeri molto grandi.

Gli algoritmi più potenti per la fattorizzazione di scopo generale sono basati sul crivello: il Quadratic Sieve (QS) e il General Number Field Sieve (GNFS). Questi algoritmi condividono una strategia comune: cercare congruenze di quadrati modulo  $n$ , cioè trovare interi  $x$  e  $y$  tali che  $x^2 \equiv y^2 \pmod{n}$  ma  $x \not\equiv \pm y \pmod{n}$ . Se tale congruenza esiste, allora  $n$  divide  $(x-y)(x+y)$  ma  $n$  non divide né  $x-y$  né  $x+y$  separatamente, quindi  $\gcd(x-y, n)$  e  $\gcd(x+y, n)$  rivelano fattori non banali di  $n$ .

Il Quadratic Sieve cerca relazioni della forma  $Q(a) = (a + L\sqrt{n})^2 - n$  che siano smooth rispetto a una base di fattori scelta. Ogni relazione smooth fornisce un'equazione  $Q(a) = \prod p_i^{e_i}$  dove i  $p_i$  sono primi nella base di fattori. Raccogliendo molte tali relazioni e usando algebra lineare su  $F_2$  per trovare sottoinsiemi il cui prodotto sia un quadrato perfetto, si costruiscono le congruenze necessarie. La complessità del Quadratic Sieve è euristica  $L_n[1/2, 1] = \exp((1 + o(1)) \cdot \sqrt{\log n \cdot \log \log n})$ .

Il General Number Field Sieve rappresenta attualmente l'algoritmo di fattorizzazione più veloce conosciuto per interi generali grandi. La sua complessità euristica è  $L_n[1/3, (64/9)^{1/3}] \approx \exp(1.923 \cdot (\log n)^{1/3} \cdot (\log \log n)^{2/3})$ . GNFS è notevolmente più veloce di QS per moduli crittograficamente rilevanti: per un modulo a 1024 bit, QS richiederebbe circa  $2^{86}$  operazioni mentre GNFS richiede circa  $2^{64}$  operazioni. Per un modulo a 2048 bit, le stime sono circa  $2^{116}$  per QS e  $2^{96}$  per GNFS.

L'algoritmo GNFS lavora in un campo numerico  $Q(\alpha)$  dove  $\alpha$  è una radice di un polinomio irriducibile  $f(x)$  di grado  $d$ , scelto in modo che  $f(m) \equiv 0 \pmod{n}$  per qualche intero  $m$  vicino a  $n^{1/d}$ . L'algoritmo cerca coppie  $(a, b)$  tali che sia  $a - b\alpha$  che la norma  $N(a - b\alpha)$  dal campo numerico a  $Q$  siano smooth rispetto a basi di fattori appropriate. Queste relazioni permettono di costruire congruenze di quadrati in  $Z_n$  mediante l'omomorfismo  $Q(\alpha) \rightarrow Z_n$  dato da  $\alpha \mapsto$

m. La fase di crivello, che domina la complessità, richiede di testare miliardi o triloni di coppie (a, b) per trovare quelle che producono valori smooth.

Record di fattorizzazione sono stati stabiliti usando GNFS su cluster di computer massivamente paralleli. Il record attuale per la fattorizzazione di un numero RSA challenge è RSA-250, un numero di 250 cifre decimali (829 bit), fattorizzato nel 2020 dopo circa 2700 anni-core di calcolo. Questi record dimostrano che moduli RSA inferiori a 1024 bit sono praticamente vulnerabili con risorse sufficienti. Gli standard moderni raccomandano moduli di almeno 2048 bit, con 3072 o 4096 bit per sicurezza a lungo termine.

## L'Algoritmo di Shor e la Minaccia Quantistica

Nel 1994, Peter Shor presentò un algoritmo quantistico che rivoluzionò completamente il panorama della sicurezza della fattorizzazione. L'algoritmo di Shor può fattorizzare un intero  $n$  in tempo polinomiale  $O((\log n)^3)$  su un computer quantistico sufficientemente potente. Questo risultato teorico implica che, se e quando computer quantistici scalabili diventeranno disponibili, l'intera infrastruttura crittografica basata su RSA diventerà insicura.

L'algoritmo di Shor riduce la fattorizzazione al problema di trovare il periodo di una funzione. Dato  $n$  da fattorizzare, si sceglie un intero casuale  $a$  coprimo con  $n$ . L'algoritmo quantistico trova il periodo  $r$  della funzione  $f(x) = a^x \bmod n$ , cioè il più piccolo intero positivo  $r$  tale che  $a^r \equiv 1 \pmod{n}$ . Una volta trovato  $r$ , se  $r$  è pari e  $a^{r/2} \not\equiv -1 \pmod{n}$ , allora  $\gcd(a^{r/2} - 1, n)$  e  $\gcd(a^{r/2} + 1, n)$  sono fattori non banali di  $n$ . Se  $r$  è dispari o  $a^{r/2} \equiv -1 \pmod{n}$ , si riprova con un diverso valore di  $a$ . La probabilità di successo per ogni tentativo è almeno  $1/2$ , quindi pochi tentativi sufficienti.

La parte quantistica dell'algoritmo usa la Quantum Fourier Transform (QFT) per trovare il periodo  $r$ . Si prepara uno stato quantistico che è una sovrapposizione uniforme su tutti i valori  $x$  in un range appropriato, si applica  $f(x) = a^x \bmod n$  a questa sovrapposizione ottenendo uno stato entangled, e si misura la seconda componente. Questa misurazione collassa lo stato in una sovrapposizione periodica con periodo  $r$ . Applicando la QFT a questo stato, si ottiene un picco nelle frequenze che sono multipli di  $1/r$ , permettendo di estrarre  $r$  con alta probabilità.

La complessità temporale  $O((\log n)^3)$  dell'algoritmo di Shor include il costo di implementare l'esponenziazione modulare e la QFT su un computer quantistico. L'esponenziazione  $a^x \bmod n$  può essere implementata quantisticamente usando  $O(\log n)$  gate quantistici per ogni bit di  $x$ . La QFT su  $N$  elementi richiede  $O((\log N)^2)$  gate quantistici. Combinando questi componenti, l'algoritmo completo richiede  $O((\log n)^3)$  gate quantistici, un miglioramento esponenziale rispetto a GNFS.

Tuttavia, costruire un computer quantistico con abbastanza qubit e bassa abbastanza correzione d'errore per eseguire l'algoritmo di Shor su moduli crittograficamente rilevanti rimane una sfida ingegneristica formidabile. Stime conservative suggeriscono che fattorizzare un modulo RSA a 2048 bit richiederebbe circa 20 milioni di qubit fisici con tassi di errore attuali, o circa 4000 qubit logici con correzione d'errore perfetta. I computer quantistici attuali hanno poche centinaia di qubit con tassi di errore ancora troppo alti per calcoli complessi.

Nonostante queste limitazioni pratiche attuali, la minaccia teorica dell'algoritmo di Shor è sufficiente per motivare la transizione alla crittografia post-quantistica. Le agenzie di sicurezza nazionali raccomandano di pianificare la migrazione da RSA e altri schemi vulnerabili a Shor verso schemi resistenti ai computer quantistici. Il National Institute of Standards and Technology (NIST) ha condotto un processo di standardizzazione multi-anno per identificare e standardizzare algoritmi post-quantistici, con i primi standard rilasciati recentemente.

## Il Sistema Crittografico RSA e le sue Varianti

Il sistema RSA, inventato da Rivest, Shamir e Adleman nel 1978, è costruito come segue. Si scelgono due primi distinti grandi  $p$  e  $q$ , tipicamente di lunghezza comparabile, e si calcola il modulo  $n = p \cdot q$ . Si calcola la funzione totiente di Eulero  $\phi(n) = (p-1)(q-1)$ , che dà l'ordine del gruppo moltiplicativo  $(\mathbb{Z}_n)^*$ . Si sceglie un esponente di cifratura  $e$  tale che  $\gcd(e, \phi(n)) = 1$ , tipicamente  $e = 65537 = 2^{16} + 1$  per efficienza. Si calcola l'esponente di decifratura  $d$  tale che  $e \cdot d \equiv 1 \pmod{\phi(n)}$ , usando l'algoritmo euclideo esteso. La chiave pubblica è  $(n, e)$  e la chiave privata è  $(n, d)$ , o più spesso  $(p, q, d)$  per permettere ottimizzazioni come il teorema cinese del resto.

Per cifrare un messaggio  $m \in \mathbb{Z}_n$ , si calcola il testo cifrato  $c = m^e \bmod n$ . Per decifrare, si calcola  $m = c^d \bmod n$ . La correttezza segue dal teorema di Eulero: se  $\gcd(m, n) = 1$ , allora  $m^{\phi(n)} \equiv 1 \pmod{n}$ , quindi  $c^d = (m^e)^d = m^{ed} = m^{1 + k \cdot \phi(n)} = m \cdot (m^{\phi(n)})^k \equiv m \pmod{n}$  per qualche intero  $k$ .

La sicurezza di RSA si basa sulla congettura RSA, che asserisce che invertire la funzione  $f(m) = m^e \bmod n$  è computazionalmente difficile senza conoscere la fattorizzazione di  $n$ . È immediato osservare che se possiamo fattorizzare  $n$ , possiamo rompere RSA: dalla fattorizzazione  $n = p \cdot q$ , calcoliamo  $\phi(n) = (p-1)(q-1)$ , poi calcoliamo  $d = e^{-1} \bmod \phi(n)$ , e possiamo decifrare qualsiasi testo cifrato. La questione aperta è se rompere RSA sia equivalente a fattorizzare  $n$ , o se esista un modo di invertire la funzione RSA senza fattorizzare.

Questa questione è stata parzialmente risolta in alcune direzioni. Boneh e Venkatesan mostrarono che se esiste un oracolo che, dato un testo cifrato  $c$ , rivela una frazione sufficiente dei bit del messaggio  $m$ , allora questo oracolo può essere usato per fattorizzare  $n$  in tempo polinomiale. Questo risultato suggerisce che RSA ha una proprietà di "all-or-nothing": o il messaggio è completamente nascosto, o può essere completamente recuperato. Tuttavia, la questione generale se calcolare  $d$  da  $(n, e)$  sia equivalente a fattorizzare  $n$  rimane aperta.

Un problema correlato è il calcolo di radici  $e$ -esime modulo  $n$ . Dato  $y$  e  $e$ , trovare  $x$  tale che  $x^e \equiv y \pmod{n}$  è esattamente il problema di decifratura RSA. Se  $e = 2$  e  $n = p \cdot q$  con  $p, q \equiv 3 \pmod{4}$ , estrarre radici quadrate è equivalente a fattorizzare: Rabin mostrò che un algoritmo che calcola radici quadrate modulo  $n$  può essere usato per fattorizzare  $n$  con alta probabilità. La riduzione funziona come segue: dato  $n$ , si sceglie un quadrato casuale  $y = x^2 \bmod n$  e si usa l'algoritmo per trovare una radice quadrata  $z$  di  $y$ . Con probabilità  $1/2$ ,  $z \not\equiv \pm x \pmod{n}$ , e in questo caso  $\gcd(x-z, n)$  è un fattore non banale. Questo risultato di Rabin è

uno dei pochi casi in cui abbiamo un'equivalenza dimostrabile tra rompere uno schema crittografico e risolvere un problema computazionale ben definito.

Per esponenti più grandi come  $e = 65537$ , non è nota un'equivalenza simile. La congettura prevalente è che rompere RSA con tali esponenti sia equivalente a fattorizzare, ma questa equivalenza non è stata dimostrata. Questa incertezza teorica è una delle ragioni per cui si studiano schemi alternativi con riduzioni di sicurezza più strette a problemi ben compresi.

RSA può essere usato anche per firme digitali. Una firma su un messaggio  $m$  è calcolata come  $s = m^d \bmod n$  usando la chiave privata. La verifica calcola  $m' = s^e \bmod n$  e controlla se  $m' = m$ . Se le componenti sono correttamente correlate, la verifica ha successo. In pratica, si firma l'hash del messaggio piuttosto che il messaggio stesso, per efficienza e sicurezza. Lo schema completo include padding crittografico come PSS (Probabilistic Signature Scheme) per prevenire attacchi.

Un problema sottile con RSA è la scelta dei primi  $p$  e  $q$ . Non è sufficiente che siano semplicemente grandi e casuali. Devono soddisfare varie condizioni di sicurezza per prevenire attacchi specifici. Ad esempio,  $p-1$  e  $q-1$  devono entrambi avere almeno un fattore primo grande per resistere al metodo  $p-1$  di Pollard. La differenza  $|p-q|$  non deve essere troppo piccola, altrimenti l'attacco di fattorizzazione di Fermat funziona: se  $p$  e  $q$  sono vicini, allora  $n \approx p^2$ , quindi  $\sqrt{n} \approx p$ , e provando valori vicini a  $\sqrt{n}$  si trova rapidamente  $p$ . I primi  $p$  e  $q$  non devono essere scelti da distribuzioni con pattern prevedibili. Varie implementazioni RSA sono state compromesse a causa di generazione debole di primi, come l'incidente dove molti dispositivi embedded generarono primi da entropia insufficiente, producendo moduli che condividevano fattori comuni.

## Connessioni con Altri Problemi Computazionali

Il problema della fattorizzazione ha connessioni profonde con altri problemi computazionali in teoria dei numeri e complessità. Un problema strettamente correlato è il calcolo della funzione totiente  $\phi(n)$ . Dato  $n$ , calcolare  $\phi(n)$  è equivalente a fattorizzare  $n$ . Infatti, se conosciamo  $\phi(n)$  e  $n$ , possiamo fattorizzare  $n$  risolvendo il sistema  $p + q = n - \phi(n) + 1$  e  $p \cdot q = n$ , che dà un'equazione quadratica in  $p$ . Inversamente, se conosciamo la fattorizzazione  $n = p_1^{a_1} \cdot \dots \cdot p_k^{a_k}$ , calcoliamo  $\phi(n) = \prod p_i^{a_i-1} \cdot (p_i - 1)$ .

Un altro problema equivalente è il calcolo dell'ordine di un elemento in  $(\mathbb{Z}_n)^*$ . Se *possiamo calcolare ordini in  $(\mathbb{Z}_n)^*$*  per elementi casuali, possiamo fattorizzare  $n$ . L'idea è che l'ordine di un elemento casuale tende a rivelare informazioni sulla struttura di  $(\mathbb{Z}_n)^*$ , che a sua volta rivela i fattori primi di  $n$ .

Il problema del residuo quadratico chiede se un dato elemento  $a \in (\mathbb{Z}_n)^*$  è un quadrato, cioè se esiste  $x$  tale che  $x^2 \equiv a \pmod{n}$ . Decidere questo problema è equivalente a fattorizzare  $n$ . Se  $n = p \cdot q$  è noto, il calcolo è facile usando il simbolo di Legendre. Ma senza conoscere la fattorizzazione, il problema è difficile. Goldwasser e Micali costruirono uno schema di cifratura probabilistico basato su questa durezza, chiamato cifratura a residui quadratici.

La fattorizzazione è anche connessa al problema del logaritmo discreto in modi sottili. Per il gruppo  $(\mathbb{Z}_n)^*$  con  $n$  composto, il logaritmo discreto può essere ridotto al logaritmo discreto

nei sottogruppi primi  $(\mathbb{Z}_p)^*$  e  $(\mathbb{Z}_q)^*$  usando il teorema cinese del resto, assumendo di conoscere la fattorizzazione  $n = p \cdot q$ . Inversamente, non è noto se la capacità di calcolare logaritmi discreti in  $(\mathbb{Z}_n)^*$  senza conoscere la fattorizzazione permetta di fattorizzare  $n$ . Questa asimmetria è interessante: la fattorizzazione sembra aiutare con il logaritmo discreto, ma non viceversa.

## Parte III: Reticoli e Crittografia Post-Quantistica

### Il Problema del Vettore Più Corto e le sue Varianti

La crittografia basata su reticoli emerge come il candidato principale per la sicurezza post-quantistica, resistendo sia agli algoritmi classici che quantistici noti. Un reticolo è un insieme discreto di punti nello spazio euclideo  $\mathbb{R}^n$  ottenuto come combinazione lineare intera di vettori base linearmente indipendenti. Formalmente, dato un insieme di vettori base  $B = \{b_1, b_2, \dots, b_m\}$  in  $\mathbb{R}^n$  con  $m \leq n$ , il reticolo generato da  $B$  è:

$$L(B) = \{\sum_{i=1}^m z_i \cdot b_i : z_i \in \mathbb{Z}\}$$

Il problema del vettore più corto (Shortest Vector Problem, SVP) chiede di trovare un vettore non nullo  $v \in L$  tale che la norma euclidea  $\|v\|$  sia minima tra tutti i vettori non nulli del reticolo. La distanza minima  $\lambda_1(L)$  è definita come  $\lambda_1(L) = \min\{\|v\| : v \in L, v \neq 0\}$ . Il problema SVP chiede di trovare un vettore  $v$  con  $\|v\| = \lambda_1(L)$ .

Varianti del problema SVP includono l'approssimazione: dato un fattore  $\gamma \geq 1$ , trovare un vettore non nullo  $v \in L$  tale che  $\|v\| \leq \gamma \cdot \lambda_1(L)$ . Il problema approssimato è denotato  $\gamma$ -SVP. Per  $\gamma = 1$ , otteniamo il problema esatto. Per  $\gamma$  polinomiale o esponenziale nella dimensione  $n$ , otteniamo varianti progressivamente più facili ma ancora congetturalmente difficili.

Un altro problema fondamentale è il problema del vettore più vicino (Closest Vector Problem, CVP). Dato un reticolo  $L$  e un punto target  $t \in \mathbb{R}^n$  non necessariamente nel reticolo, CVP chiede di trovare il vettore  $v \in L$  che minimizza la distanza  $\|t - v\|$ . Anche CVP ha versioni approssimate  $\gamma$ -CVP che chiedono di trovare  $v \in L$  con  $\|t - v\| \leq \gamma \cdot \text{dist}(t, L)$  dove  $\text{dist}(t, L) = \min\{\|t - w\| : w \in L\}$ .

I risultati di durezza computazionale per questi problemi sono complessi e dipendono fortemente dalla dimensione del reticolo, dal fattore di approssimazione  $\gamma$ , e dalla norma utilizzata. Ajtai dimostrò nel 1996 un risultato rivoluzionario: SVP è NP-hard sotto riduzioni randomizzate. Più precisamente, mostrò che  $\gamma$ -SVP per  $\gamma = n^{c/\log \log n}$  (una funzione che cresce più lentamente di qualsiasi polinomio) è NP-hard sotto riduzioni probabilistiche. Risultati successivi di Micciancio e altri hanno rafforzato questo, mostrando che  $\gamma$ -SVP è NP-hard per  $\gamma$  quasi-polinomiale.

Per CVP, la situazione è ancora più forte: CVP esatto è NP-hard sotto riduzioni deterministiche, dimostrato originariamente da van Emde Boas. Inoltre,  $\gamma$ -CVP per  $\gamma = n^{c/\log n}$  è NP-hard. Questi risultati di NP-difficoltà suggeriscono che i problemi su reticoli sono intrinsecamente difficili nel caso peggiore (worst-case).

Tuttavia, la crittografia non può basarsi su durezza worst-case. I parametri crittografici devono garantire che il problema sia difficile in media per istanze scelte casualmente, non solo nel caso peggiore. La svolta per la crittografia su reticoli arrivò con i risultati di Ajtai e successivamente di Regev, che mostrarono riduzioni dal caso peggiore al caso medio per certi problemi su reticoli. Questi risultati stabiliscono che se possiamo risolvere efficientemente certe varianti del problema su reticoli per istanze casuali, allora possiamo risolvere efficientemente SVP o CVP per tutte le istanze, incluse quelle peggiori.

Le riduzioni worst-case to average-case rappresentano una proprietà unica e potente dei problemi su reticoli, non condivisa dalla fattorizzazione o dal logaritmo discreto. Questa proprietà fornisce una giustificazione teorica più forte per la sicurezza crittografica: anche se la durezza worst-case di SVP non garantisce direttamente la sicurezza crittografica, la riduzione implica che rompere lo schema crittografico permetterebbe di risolvere istanze worst-case di SVP, che è congetturalmente impossibile.

## Algoritmi per Problemi su Reticoli

Gli algoritmi per SVP e CVP sono significativamente diversi da quelli per fattorizzazione o logaritmo discreto. L'algoritmo più semplice per SVP è l'enumerazione completa: esplorare sistematicamente la palla centrata nell'origine fino a trovare il vettore più corto del reticolo. La complessità è essenzialmente  $2^{O(n)}$ , esponenziale nella dimensione. Per reticoli di dimensione crittograficamente rilevante ( $n > 500$ ), questo è completamente infattibile.

L'algoritmo LLL (Lenstra-Lenstra-Lovász), sviluppato nel 1982, fornisce un'approssimazione polinomiale per SVP. LLL trova in tempo polinomiale  $O(n^6 \cdot \log B)$  un vettore  $v$  con  $\|v\| \leq 2^{(n/2)} \cdot \lambda_1(L)$ , dove  $B$  è un bound sulla lunghezza dei vettori base iniziali. Il fattore di approssimazione esponenziale  $2^{(n/2)}$  rende LLL inadatto per rompere schemi crittografici su reticoli con dimensioni sufficientemente grandi, ma l'algoritmo è fondamentale per analisi di sicurezza e per costruire primitive crittografiche.

LLL opera riducendo la base del reticolo mediante una combinazione di ortogonalizzazione di Gram-Schmidt e riduzioni. L'algoritmo mantiene invarianti che garantiscono che i vettori base rimangano "quasi ortogonali" in un senso tecnico. Il risultato è una base ridotta da cui si può estrarre un vettore relativamente corto. L'algoritmo LLL ha applicazioni vastissime oltre la crittografia, inclusa la risoluzione di equazioni diofantee, l'attacco a schemi crittografici deboli, e problemi in teoria dei numeri computazionale.

Algoritmi più sofisticati come BKZ (Block Korkine-Zolotarev) migliorano l'approssimazione di LLL a costo di tempo esponenziale. BKZ con blocco di dimensione  $\beta$  fornisce fattori di approssimazione circa  $(\beta)^{n/(2\beta)}$ , operando in tempo esponenziale in  $\beta$ . Scegliendo  $\beta$  appropriatamente, si può bilanciare qualità dell'approssimazione e tempo computazionale. Nella pratica, implementazioni ottimizzate di BKZ con blocchi di dimensione  $\beta \approx 50-100$  sono lo stato dell'arte per attaccare schemi crittografici su reticoli.

Un altro approccio è il setacciamento (sieving), che esplora sistematicamente lo spazio dei vettori corti combinandoli per produrre vettori ancora più corti. Gli algoritmi di sieving classici hanno complessità  $2^{cn}$  per qualche costante  $c$ , tipicamente  $c \approx 0.3$ . Algoritmi quantistici di sieving, proposti da Laarhoven e altri, raggiungono complessità circa  $2^{(0.265n)}$ , un miglioramento polinomiale ma non esponenziale rispetto agli algoritmi classici. Questo

suggerisce che i computer quantistici offrono solo vantaggi limitati contro la crittografia su reticoli, a differenza del caso della fattorizzazione e del logaritmo discreto dove gli algoritmi quantistici danno vantaggi esponenziali.

La stima della sicurezza concreta per schemi crittografici su reticoli richiede di modellare attentamente la complessità di questi algoritmi per parametri specifici. Strumenti come il Lattice Estimator forniscono stime basate su modelli euristici del comportamento di BKZ e sieving per diverse dimensioni e parametri di reticolo. Questi modelli guidano la scelta di parametri crittografici: ad esempio, per sicurezza a 128 bit (equivalente a  $2^{128}$  operazioni per rompere), si scelgono tipicamente reticoli di dimensione  $n \approx 512-768$  con parametri di rumore appropriati.

## Il Problema Learning With Errors (LWE)

Il problema Learning With Errors, introdotto da Oded Regev nel 2005, ha rivoluzionato la crittografia su reticoli fornendo una base più flessibile e versatile per costruire schemi crittografici. LWE è definito come segue: sia  $n$  un intero (la dimensione),  $q$  un modulo (tipicamente primo), e  $\chi$  una distribuzione di errore su  $\mathbb{Z}_q$  (tipicamente una distribuzione discreta gaussiana con deviazione standard piccola  $\sigma$ ).

Il problema decisionale LWE chiede di distinguere tra due distribuzioni:

1. Campioni LWE: coppie  $(a_i, b_i)$  dove  $a_i$  è scelto uniformemente da  $\mathbb{Z}_q^n$ ,  $s$  è un vettore segreto fisso in  $\mathbb{Z}_q^n$ ,  $e_i$  è un errore piccolo campionato da  $\chi$ , e  $b_i = \langle a_i, s \rangle + e_i \bmod q$ .
2. Campioni uniformi: coppie  $(a_i, b_i)$  dove sia  $a_i$  che  $b_i$  sono scelti uniformemente e indipendentemente da  $\mathbb{Z}_q^n$  e  $\mathbb{Z}_q$  rispettivamente.

Un algoritmo risolve decisional-LWE se, dati  $m$  campioni da una delle due distribuzioni, può determinare con probabilità significativamente maggiore di  $1/2$  da quale distribuzione provengono i campioni.

Il problema search-LWE chiede, dati campioni  $(a_i, b_i = \langle a_i, s \rangle + e_i)$ , di recuperare il vettore segreto  $s$ . Regev mostrò che search-LWE e decisional-LWE sono equivalenti mediante riduzioni polinomiali, quindi dal punto di vista della complessità sono essenzialmente lo stesso problema.

La svolta fondamentale di Regev fu dimostrare una riduzione dal caso peggiore al caso medio: mostrò che risolvere LWE in media (per parametri opportuni) è difficile quanto risolvere  $\gamma$ -SVP nel caso peggiore per  $\gamma = \tilde{O}(n/\sigma)$ . Questa riduzione fornisce una giustificazione teorica forte per la durezza di LWE: se LWE fosse facile in media, allora SVP sarebbe facile nel caso peggiore, contraddicendo la congettura di durezza di SVP.

La riduzione di Regev è quantistica: l'algoritmo che riduce SVP worst-case a LWE average-case richiede un computer quantistico. Successivamente, Peikert e altri hanno dimostrato riduzioni classiche simili con parametri leggermente peggiori. L'esistenza di

riduzioni sia classiche che quantistiche suggerisce che LWE cattura una durezza fondamentale dei problemi su reticoli.

Varianti di LWE includono Ring-LWE e Module-LWE, che lavorano con reticoli strutturati algebricamente piuttosto che reticoli completamente casuali. Ring-LWE, introdotto da Lyubashevsky, Peikert e Regev, opera nell'anello dei polinomi  $R_q = \mathbb{Z}_q[x]/(f(x))$  per qualche polinomio irriducibile  $f(x)$ , tipicamente  $f(x) = x^n + 1$  con  $n$  una potenza di 2. Gli elementi sono polinomi di grado minore di  $n$ , e le operazioni sono eseguite modulo  $f(x)$  e modulo  $q$ .

Il problema Ring-LWE è analogo a LWE ma opera in  $R_q$  invece di  $\mathbb{Z}_q^n$ . I campioni sono coppie  $(a_i, b_i = a_i \cdot s + e_i)$ , dove  $a_i$  è scelto uniformemente da  $R_q$ ,  $s$  è il segreto in  $R_q$ , ed  $e_i$  è un errore piccolo estratto da una distribuzione di errore (tipicamente una gaussiana discreta) su  $R_q$ . La struttura ad anello permette rappresentazioni più compatte e operazioni più efficienti grazie alla possibilità di utilizzare trasformate veloci come la Number Theoretic Transform (NTT), riducendo la complessità computazionale da  $O(n^2)$  a  $O(n \log n)$  per la moltiplicazione polinomiale.

Module-LWE generalizza ulteriormente questo concetto operando su moduli di rango  $d$  su  $R_q$ , offrendo un compromesso tra l'efficienza di Ring-LWE e la flessibilità di LWE standard. La sicurezza di queste varianti si basa sulla difficoltà computazionale di problemi su reticoli strutturati, ed è alla base di schemi crittografici post-quantistici come CRYSTALS-Kyber e CRYSTALS-Dilithium.

## Parte IV: Crittografia Basata su Codici e il Problema della Decodifica della Sindrome

### Il Problema della Decodifica e la sua difficoltà computazionale

La crittografia basata su codici correttori di errori rappresenta uno dei pilastri più antichi della crittografia post-quantistica, con radici che risalgono al crittosistema di McEliece proposto nel 1978, contemporaneamente all'avvento di RSA. Mentre RSA e altri schemi basati sulla fattorizzazione o sul logaritmo discreto sono vulnerabili agli algoritmi quantistici, i sistemi basati su codici hanno resistito sia agli attacchi classici che a quelli quantistici per oltre quattro decenni, rendendoli candidati naturali per la sicurezza nell'era post-quantistica.

Il fondamento matematico di questi sistemi risiede nella teoria dei codici lineari e, più specificamente, nella difficoltà computazionale del problema della decodifica del sindrome (Syndrome Decoding Problem). Per comprendere appieno questo problema e le sue implicazioni crittografiche, dobbiamo iniziare con una breve digressione sulla teoria dei codici correttori di errori.

Un codice lineare binario  $[n, k, d]$  è un sottospazio vettoriale  $k$ -dimensionale di  $F_2^n$ , dove  $F_2 = \{0, 1\}$  è il campo finito a due elementi. Gli elementi del codice sono chiamati parole codice, e la distanza minima  $d$  è la minima distanza di Hamming tra due parole codice distinte. La distanza di Hamming tra due vettori è il numero di posizioni in cui differiscono:  $d_H(x, y) = |\{i$

:  $x_i \neq y_i$ }. La capacità di correzione del codice è determinata dalla distanza minima: un codice con distanza minima  $d$  può correggere fino a  $\lfloor (d-1)/2 \rfloor$  errori.

Un codice lineare può essere specificato mediante una matrice generatrice  $G$  di dimensione  $k \times n$ , tale che lo spazio delle parole codice sia  $C = \{mG : m \in F_2^k\}$ . Equivalentemente, il codice può essere specificato mediante una matrice di controllo di parità  $H$  di dimensione  $(n-k) \times n$ , tale che  $C = \{c \in F_2^n : Hc^T = 0\}$ . Le due rappresentazioni sono duali: una parola  $c$  è nel codice se e solo se soddisfa  $Hc^T = 0$ .

Nel contesto della trasmissione di informazioni, un mittente codifica un messaggio  $m \in F_2^k$  come  $c = mG$  e trasmette  $c$  attraverso un canale rumoroso. Il canale introduce errori, e il destinatario riceve  $y = c + e$ , dove  $e$  è un vettore di errore con peso di Hamming  $w = |\{i : e_i = 1\}|$  relativamente piccolo. Il problema della decodifica chiede di recuperare il messaggio originale  $m$  (o equivalentemente la parola codice  $c$ ) da  $y$ .

Il síndrome di un vettore ricevuto  $y$  è definito come  $s = Hy^T$ . Poiché  $Hc^T = 0$  per ogni parola codice  $c$ , abbiamo  $s = H(c + e)^T = Hc^T + He^T = He^T$ . Il síndrome dipende solo dal vettore di errore  $e$ , non dalla parola codice trasmessa. Questo porta alla formulazione del problema della decodifica del síndrome:

**Problema della Decodifica del Síndrome (Syndrome Decoding Problem, SDP):** Dati una matrice  $H$  di dimensione  $r \times n$  su  $F_2$ , un vettore síndrome  $s \in F_2^r$ , e un intero  $w$ , trovare un vettore  $e \in F_2^n$  di peso di Hamming al più  $w$  tale che  $He^T = s$ .

Il problema generale della decodifica del síndrome è stato dimostrato essere NP-completo da Berlekamp, McEliece e van Tilborg nel 1978. Più precisamente, il problema decisionale—determinare se esiste un vettore di errore di peso al più  $w$  con un dato síndrome—è NP-completo. Questo risultato di NP-completezza fornisce una forte giustificazione teorica per la durezza del problema: se  $P \neq NP$ , allora non esiste alcun algoritmo efficiente che risolva SDP nel caso peggiore.

Tuttavia, come per i problemi su reticoli, la durezza nel caso peggiore non garantisce automaticamente la sicurezza crittografica, che richiede durezza in media per istanze scelte casualmente. La questione cruciale è se esistano distribuzioni di istanze SDP per le quali il problema rimanga difficile anche quando la matrice  $H$  e il peso  $w$  sono scelti appropriatamente. La congettura sottostante alla crittografia basata su codici asserisce che per codici casuali (codici i cui parametri non hanno struttura speciale sfruttabile) e per pesi di errore appropriati, SDP rimane computazionalmente intrattabile.

## Algoritmi per la Decodifica del Síndrome

Comprendere gli algoritmi esistenti per SDP è essenziale per valutare la sicurezza concreta dei crittosistemi basati su codici. Gli algoritmi si dividono in due categorie principali: algoritmi algebrici che sfruttano la struttura del codice specifico, e algoritmi combinatori generali che trattano il codice come una "scatola nera".

Per codici con struttura algebrica speciale, come i codici Reed-Solomon, Goppa, o BCH, esistono algoritmi di decodifica efficienti basati su tecniche algebriche. L'algoritmo di Berlekamp-Welch per i codici Reed-Solomon, ad esempio, può correggere fino a  $\lfloor (d-1)/2 \rfloor$

errori in tempo polinomiale. Questi algoritmi sfruttano l'equazione chiave e trovano i locatori degli errori risolvendo sistemi di equazioni polinomiali.

Per la crittografia, però, l'interesse risiede in codici per i quali questi algoritmi algebrici non funzionano, o funzionano solo mascherando la struttura algebrica sottostante. Nel crittosistema di McEliece, ad esempio, si usa un codice Goppa, ma la chiave pubblica è una versione "scrambled" della matrice generatrice, ottenuta mediante moltiplicazione per matrici di permutazione e inversione casuali. La speranza è che questa trasformazione nasconda sufficientemente la struttura del codice Goppa, rendendo impossibile applicare algoritmi di decodifica algebrici efficienti.

Per codici generali senza struttura sfruttabile, l'approccio più potente è la famiglia di algoritmi Information Set Decoding (ISD). L'idea fondamentale, introdotta da Prange nel 1962 e successivamente migliorata da numerosi ricercatori, è la seguente. Supponiamo che il vettore di errore  $e$  abbia peso  $w$  e che vogliamo risolvere  $He^T = s$ . Se potessimo identificare un insieme di  $k$  posizioni nelle quali  $e$  è zero, potremmo usare queste  $k$  posizioni come "information set" per risolvere direttamente per le posizioni rimanenti dove gli errori si concentrano.

L'algoritmo di Prange procede come segue. Si sceglie casualmente un sottoinsieme  $I$  di  $k$  colonne di  $H$ , si spera che queste colonne siano linearmente indipendenti e che le corrispondenti posizioni di  $e$  siano tutte zero. Si riordina  $H$  mettendo le colonne di  $I$  nella posizione di colonne leading, ottenendo una forma  $H = [I_k | Q]$  dove  $I_k$  è una matrice identità  $k \times k$ . Il sistema  $He^T = s$  diventa  $[I_k | Q][e_1^T | e_2^T]^T = s$ , che si semplifica in  $e_1 + Qe_2^T = s$ , quindi  $e_1 = s - Qe_2^T$ . Se l'ipotesi che  $e_1 = 0$  fosse corretta, allora  $s = Qe_2^T$ , e possiamo cercare  $e_2$  di peso  $w$  enumerando tutte le  $(n-k \text{ sceglie } w)$  possibilità. Se nessuna soluzione è trovata, si riprova con un diverso information set.

La probabilità che un information set casuale sia "buono" (cioè le corrispondenti posizioni di  $e$  siano tutte zero) è:

$$P = (n-w \text{ sceglie } k) / (n \text{ sceglie } k) = [\prod_{i=0}^{k-1} (n-w-i)] / [\prod_{i=0}^{k-1} (n-i)]$$

Per parametri tipici, questa probabilità è estremamente piccola, quindi l'algoritmo deve provare un numero enorme di information set prima di trovarne uno buono.

Varianti più sofisticate di ISD riducono la complessità permettendo piccoli pesi negli information set. L'algoritmo di Lee-Brickell permette che alcune posizioni nell'information set contengano errori, tipicamente 1 o 2. L'algoritmo di Stern introduce una tecnica di "collision" che suddivide il problema in sottoproblemi più piccoli risolvibili indipendentemente, le cui soluzioni vengono poi combinate. L'algoritmo di Ball Collision Decoding di Bernstein, Lange e Peters migliora ulteriormente le tecniche di collision usando rappresentazioni probabilistiche.

La complessità asintotica di questi algoritmi ISD è stata analizzata estensivamente. Per un codice  $[n, k]$  e peso di errore  $w$ , la migliore complessità conosciuta per algoritmi ISD è approssimativamente:

$$T \approx 2^{(H_2(w/n) \cdot n - (n-k) \cdot \log_2 n)}$$

dove  $H_2(p) = -p \log_2(p) - (1-p) \log_2(1-p)$  è l'entropia binaria. Questa formula cattura il trade-off tra l'entropia degli errori e l'informazione fornita dal syndrome. Per parametri crittografici tipici, questa complessità è esponenziale e costituisce la base per la scelta di parametri sicuri.

Algoritmi quantistici per SDP offrono solo miglioramenti modesti rispetto agli algoritmi classici. Applicando l'algoritmo di ricerca di Grover, si può ottenere una accelerazione quadratica, riducendo la complessità da  $2^c$  a  $2^{(c/2)}$  per qualche costante  $c$ . Questo è un miglioramento polinomiale, non esponenziale come l'algoritmo di Shor per la fattorizzazione. Di conseguenza, la crittografia basata su codici rimane sicura nell'era quantistica aumentando appropriatamente le dimensioni dei parametri.

## Il Crittosistema di McEliece e le sue Varianti

Il crittosistema di McEliece è uno schema di cifratura a chiave pubblica costruito come segue. La chiave privata consiste di tre componenti: una matrice generatrice  $G$  di un codice  $[n, k, d]$  con capacità di correzione  $t = \lfloor (d-1)/2 \rfloor$ , una matrice di permutazione  $P$  di dimensione  $n \times n$ , e una matrice invertibile non singolare  $S$  di dimensione  $k \times k$ . Il codice sottostante è tipicamente un codice Goppa binario, che ha algoritmi di decodifica efficienti noti solo a chi conosce la struttura del codice.

La chiave pubblica è la matrice  $G_{\text{pub}} = S \cdot G \cdot P$ , che appare come la matrice generatrice di un codice casuale. Le matrici  $S$  e  $P$  "mascherano" la struttura del codice Goppa, rendendo impossibile per un attaccante usare algoritmi di decodifica algebrici.

Per cifrare un messaggio  $m \in F_2^k$ , si calcola  $c = m \cdot G_{\text{pub}} + e$ , dove  $e$  è un vettore di errore casuale di peso esattamente  $t$ . Il testo cifrato è  $c \in F_2^n$ .

Per decifrare, il destinatario calcola  $c' = c \cdot P^{-1} = m \cdot S \cdot G + e \cdot P^{-1}$ . Poiché  $P$  è una permutazione, il vettore  $e \cdot P^{-1}$  ha ancora peso  $t$ . Il destinatario applica l'algoritmo di decodifica del codice Goppa a  $c'$ , recuperando  $m \cdot S$ , e infine calcola  $m = (m \cdot S) \cdot S^{-1}$ .

La sicurezza del sistema si basa sulla congettura che decifrare un testo cifrato senza conoscere la chiave privata sia equivalente a risolvere SDP per un codice casuale. Gli attacchi noti si riducono essenzialmente ad applicare algoritmi ISD, la cui complessità determina la scelta dei parametri sicuri.

Per sicurezza classica a 128 bit, i parametri tipici del McEliece originale (con codici Goppa binari) sono  $n = 3488$ ,  $k = 2720$ ,  $t = 64$ . Questi parametri producono chiavi pubbliche estremamente grandi: circa 650 KB. Questa esplosione nelle dimensioni delle chiavi è stata storicamente il principale ostacolo all'adozione pratica di McEliece, nonostante la sua lunga storia di resistenza agli attacchi.

Numerose varianti sono state proposte per ridurre le dimensioni delle chiavi. Il crittosistema BIKE (Bit Flipping Key Encapsulation) usa codici quasi-ciclici con matrici sparse, riducendo drasticamente le dimensioni delle chiavi pubbliche a pochi KB. Tuttavia, la struttura aggiuntiva introduce nuove superfici di attacco, e l'analisi di sicurezza è più complessa. Il sistema HQC (Hamming Quasi-Cyclic) adotta un approccio simile ma con tecniche di costruzione leggermente diverse.

Il NIST, nel suo processo di standardizzazione post-quantistica, ha selezionato tre schemi basati su codici come finalisti alternativi: Classic McEliece (la versione ottimizzata del McEliece originale), BIKE, e HQC. Classic McEliece è considerato il più conservativo, con decenni di analisi di sicurezza alle spalle, ma soffre ancora delle dimensioni delle chiavi. BIKE e HQC offrono chiavi più compatte ma richiedono ulteriore scrutinio crittoanalitico.

## **Attacchi Strutturali e la Fragilità delle Varianti**

Un aspetto critico della crittografia basata su codici è che l'introduzione di struttura algebrica per ridurre le dimensioni delle chiavi può aprire vulnerabilità inaspettate. Questo principio è stato drammaticamente illustrato da numerosi break di varianti proposte.

Gli attacchi strutturali sfruttano proprietà algebriche dei codici utilizzati. Per esempio, se il codice sottostante ha una struttura di gruppo o algebra, gli attaccanti possono usare tecniche di Gröbner basis o algebra lineare per recuperare informazioni sulla chiave privata. L'attacco di Sidelnikov e Shestakov contro il crittosistema basato su codici Reed-Solomon, proposto come alternativa a McEliece, sfrutta le proprietà algebriche dei codici Reed-Solomon per recuperare la struttura del codice in tempo polinomiale.

Attacchi di distinguibilità cercano di determinare se una matrice generatrice pubblica proviene da un codice con struttura speciale (come Goppa) o da un codice completamente casuale. Se un attaccante può distinguere, può potenzialmente sfruttare la struttura identificata. L'attacco di Faugère, Gauthier, Otmani, Perret e Tillich contro varianti di McEliece basate su codici Goppa "wild" usa tecniche algebriche avanzate per distinguere e attaccare tali varianti.

La lezione fondamentale è che mentre SDP per codici completamente casuali sembra essere difficile, l'aggiunta di struttura deve essere fatta con estrema cautela. La struttura che permette efficienza e chiavi compatte è esattamente ciò che gli attaccanti cercano di sfruttare. Questo crea una tensione intrinseca nel design: vogliamo abbastanza struttura per efficienza, ma non così tanta da permettere attacchi.

## **Decodifica Oltre i Codici Lineari: Il Caso dei Codici LDPC e MDPC**

I codici LDPC (Low-Density Parity-Check) e MDPC (Moderate-Density Parity-Check) rappresentano una classe di codici con matrici di parità sparse, cioè matrici in cui la maggior parte degli elementi sono zero. Questi codici hanno eccellenti proprietà di correzione d'errore e permettono algoritmi di decodifica iterativi efficienti basati su belief propagation.

Nella crittografia, i codici QC-MDPC (Quasi-Cyclic Moderate-Density Parity-Check) sono particolarmente interessanti. La struttura quasi-ciclica permette di rappresentare la matrice di parità compattamente mediante i primi blocchi della rappresentazione ciclica, riducendo drasticamente le dimensioni delle chiavi. BIKE sfrutta esattamente questa idea.

Il problema di decodifica per codici LDPC/MDPC ha caratteristiche uniche. Gli algoritmi di decodifica iterativi (come sum-product o min-sum) non garantiscono il successo della decodifica anche quando il numero di errori è inferiore alla capacità teorica di correzione. Questo fenomeno, chiamato "decoding failure", introduce una nuova complicazione

crittografica: un attaccante può osservare quando la decodifica fallisce, e questi pattern di fallimento possono rivelare informazioni sulla chiave privata.

Attacchi basati su decoding failure sono emersi come una minaccia seria per schemi basati su LDPC/MDPC. L'attacco GJS (Guo-Johansson-Stankovski) contro alcune versioni di BIKE sfrutta le informazioni rivelate dai fallimenti di decodifica per recuperare gradualmente la chiave privata. Gli schemi moderni incorporano contromisure, come il re-encryption con parametri diversi quando si rileva un fallimento, ma queste contromisure introducono overhead computazionale.

La sicurezza a lungo termine di schemi basati su LDPC/MDPC rimane un'area di ricerca attiva. Mentre i codici Goppa di McEliece classico hanno resistito a quattro decenni di crittoanalisi, le varianti LDPC/MDPC hanno solo circa un decennio di storia. La comunità crittografica è giustamente cauta nell'adottare questi schemi per applicazioni critiche senza ulteriore maturazione.

## Parte V: Crittografia Basata su Isogenie e il Collasso di SIDH

### Curve Ellittiche, Isogenie e la Promessa della Crittografia Post-Quantistica

Le curve ellittiche hanno dominato la crittografia a chiave pubblica per decenni grazie alla loro efficienza e alla presunta durezza del logaritmo discreto su curve generali. Tuttavia, come abbiamo discusso, l'algoritmo di Shor minaccia di rendere obsoleta tutta la crittografia basata su curve ellittiche classiche nell'era quantistica. Un'idea sorprendente emersa negli ultimi due decenni è che, mentre il logaritmo discreto su curve ellittiche soccombe agli attacchi quantistici, problemi correlati ma distinti basati su *isogenie* tra curve potrebbero resistere sia ad attacchi classici che quantistici.

Un'isogenia tra due curve ellittiche  $E_1$  ed  $E_2$  definite su un campo finito  $F_q$  è un morfismo algebrico  $\varphi: E_1 \rightarrow E_2$  che preserva la struttura di gruppo e mappa il punto all'infinito in sé stesso. Le isogenie sono caratterizzate dal loro grado, che è la dimensione del kernel di  $\varphi$ . Due curve sono isogene se esiste un'isogenia tra loro, e la relazione di isogenia è una relazione di equivalenza che partiziona l'insieme di tutte le curve ellittiche su  $F_q$  in classi di isogenia.

Il grafo delle isogenie è un oggetto matematico ricco e profondo. I vertici sono curve ellittiche su  $F_q$ , e esiste un arco da  $E_1$  a  $E_2$  se esiste un'isogenia di grado primo  $\ell$  tra loro. Per un grado  $\ell$  fissato, questo grafo ha proprietà notevoli: è connesso (tutte le curve nella stessa classe di isogenia sono collegate), è regolare (ogni vertice ha grado  $\ell + 1$ ), e ha struttura di espansore (ha buone proprietà di connettività e mixing).

Il problema computazionale fondamentale per la crittografia basata su isogenie è il seguente:

**Problema del Cammino nel Grafo delle Isogenie (Isogeny Path Problem):** Date due curve ellittiche  $E_1$  ed  $E_2$  che sono isogene, trovare un'isogenia (o una sequenza di isogenie di grado piccolo) da  $E_1$  a  $E_2$ .

Questo problema è congettzionalmente difficile sia classicamente che quantisticamente. Gli algoritmi classici noti hanno complessità esponenziale nella lunghezza del cammino cercato. Gli algoritmi quantistici, incluse varianti subottimali dell'algoritmo di Shor, offrono al massimo accelerazioni subquadratiche, lasciando il problema essenzialmente esponenziale.

La congettura di durezza delle isogenie afferma che per curve ellittiche ordinarie su  $F_p$  con  $p$  primo grande e per gradi di isogenia appropriatamente grandi, il problema del cammino rimane computazionalmente intrattabile. Questa congettura è supportata da decenni di ricerca in teoria dei numeri computazionale e geometria aritmetica, dove le isogenie sono state studiate estensivamente senza portare ad algoritmi efficienti per problemi generali.

## Il Protocollo SIDH: Un'Idea Elegante

Il protocollo Supersingular Isogeny Diffie-Hellman (SIDH), proposto da De Feo, Jao e Plût nel 2011, rappresentava uno dei candidati più promettenti per lo scambio di chiavi post-quantistico. L'idea centrale è una traduzione del protocollo classico Diffie-Hellman nel linguaggio delle isogenie su curve ellittiche supersingolari.

Le curve supersingolari hanno proprietà speciali che le rendono particolarmente adatte per la crittografia basata su isogenie. In particolare, il grafo delle isogenie di grado primo  $\ell$  su curve supersingolari ha struttura particolarmente regolare e le computazioni con isogenie sono più efficienti che per curve ordinarie.

Il setup di SIDH inizia con una curva supersingolare pubblica  $E_0$  su  $F_{\{p^2\}}$  dove  $p$  è un primo della forma  $p = 2^a \cdot 3^b - 1$  per interi  $a, b$ . La curva ha ordine  $(p+1)^2 = (2^a \cdot 3^b)^2$ . Alice e Bob concordano su punti base  $P_A, Q_A$  che generano il sottogruppo di ordine  $2^a$ , e punti  $P_B, Q_B$  che generano il sottogruppo di ordine  $3^b$ .

Per eseguire lo scambio di chiavi:

1. Alice sceglie due scalari segreti  $m_A, n_A$  e computa il punto  $R_A = [m_A]P_A + [n_A]Q_A$ . Il sottogruppo generato da  $R_A$  definisce un'isogenia  $\phi_A: E_0 \rightarrow E_A$  di grado  $2^a$ .
2. Alice computa la curva immagine  $E_A = \phi_A(E_0)$  e le immagini  $\phi_A(P_B), \phi_A(Q_B)$  dei punti base di Bob. Invia la tripla  $(E_A, \phi_A(P_B), \phi_A(Q_B))$  a Bob.
3. Bob esegue il procedimento simmetrico scegliendo  $m_B, n_B$ , computando  $R_B = [m_B]P_B + [n_B]Q_B$ , e l'isogenia  $\phi_B: E_0 \rightarrow E_B$  di grado  $3^b$ . Invia  $(E_B, \phi_B(P_A), \phi_B(Q_A))$  ad Alice.
4. Alice usa l'informazione ricevuta da Bob per costruire l'isogenia  $\phi'_A: E_B \rightarrow E_{AB}$  con lo stesso kernel di  $\phi_A$  ma dominio  $E_B$  invece di  $E_0$ .

5. Bob fa similmente, costruendo  $\varphi_B: E_A \rightarrow E\{BA\}$ .
6. Per commutatività delle isogenie con kernel che commutano,  $E_{\{AB\}} = E_{\{BA\}}$ . Questa curva condivisa diventa la base per il segreto condiviso.

L'eleganza di SIDH risiede nella sua analogia diretta con Diffie-Hellman classico. Le isogenie  $\varphi_A$  e  $\varphi_B$  giocano il ruolo degli esponenti segreti  $a$  e  $b$ , e la curva  $E_{\{AB\}}$  corrisponde al segreto condiviso  $g^{ab}$ . La differenza cruciale è che mentre l'algoritmo di Shor rompe il logaritmo discreto quantisticamente, non sembrava applicabile alle isogenie su curve supersingolari.

Le dimensioni delle chiavi di SIDH erano notevolmente compatte per standard post-quantistici: circa 330 byte per sicurezza quantistica a 128 bit, comparabili a schemi di curve ellittiche classiche e molto più piccole di schemi basati su reticoli o codici. Questa compattezza, combinata con la presunta sicurezza quantistica, rendeva SIDH estremamente attraente per applicazioni con limitazioni di banda o storage.

## Il Collasso Improvviso: L'Attacco di Castryck-Decru (2022)

Nel luglio 2022, la comunità crittografica fu scossa dall'annuncio di un attacco devastante contro SIDH da parte di Wouter Castryck e Thomas Decru. L'attacco, che recupera la chiave privata in tempo polinomiale, ha completamente distrutto la sicurezza di SIDH e del suo derivato SIKE (Supersingular Isogeny Key Encapsulation), che era un finalista di terzo round nel processo di standardizzazione post-quantistica del NIST.

L'attacco sfrutta una caratteristica specifica di SIDH che era stata inclusa per ragioni di efficienza: la trasmissione dei punti ausiliari  $\varphi_A(P_B)$ ,  $\varphi_A(Q_B)$  insieme alla curva  $E_A$ . Questi punti extra forniscono informazione sufficiente per un attaccante per recuperare l'isogenia segreta  $\varphi_A$  usando tecniche sofisticate di teoria dei numeri algebrica.

L'idea chiave dell'attacco coinvolge la teoria di Kani. Dato  $(E_A, \varphi_A(P_B), \varphi_A(Q_B))$ , un attaccante può costruire una corrispondenza speciale tra  $E_0 \times E_A$  chiamata Kani endomorphism. Questa corrispondenza codifica informazione sull'isogenia  $\varphi_A$  in un modo che può essere estratto algebricamente. Lavorando con l'anello degli endomorfismi della superficie abeliana  $E_0 \times E_A$  e usando algoritmi per fattorizzare corrispondenze in prodotti di isogenie, l'attaccante può essenzialmente "leggere" l'isogenia segreta dalla traccia lasciata sui punti ausiliari.

Più tecnicamente, l'attacco procede costruendo un grafico di isogenie ausiliario che connette diverse curve correlate attraverso isogenie di grado piccolo. L'informazione fornita dai punti  $\varphi_A(P_B)$  e  $\varphi_A(Q_B)$  permette di navigare questo grafico in modo mirato, eliminando progressivamente candidati per il kernel di  $\varphi_A$  fino a quando rimane solo una possibilità. Il processo richiede tempo polinomiale, tipicamente poche ore su hardware moderno per parametri SIDH standard che erano progettati per fornire sicurezza quantistica a 128 bit.

La velocità e completezza del collasso di SIDH sono state sorprendenti. Entro settimane dall'annuncio iniziale, implementazioni dell'attacco erano disponibili pubblicamente e SIKE fu ritirato dalla competizione NIST. L'episodio illustra drammaticamente quanto possa essere

fragile la sicurezza crittografica quando si basa su problemi computazionali non completamente compresi.

## **Lezioni dall'Episodio SIDH e Direzioni Future**

Il collasso di SIDH offre lezioni profonde per la crittografia post-quantistica e, più in generale, per il design di sistemi crittografici basati su congetture non dimostrate.

Prima lezione: l'aggiunta di struttura o informazione ausiliaria per migliorare l'efficienza può introdurre vulnerabilità catastrofiche. Nel caso di SIDH, i punti ausiliari  $\phi_A(P_B)$ ,  $\phi_A(Q_B)$  erano inclusi precisamente per permettere al destinatario di computare efficientemente il segreto condiviso. Questa stessa informazione si è rivelata fatale per la sicurezza. Il principio generale è che ogni bit di informazione aggiuntiva fornito all'attaccante deve essere giustificato da un'analisi di sicurezza rigorosa, e anche allora rimangono rischi di attacchi non previsti.

Seconda lezione: la giovinezza di una primitiva crittografica è un fattore di rischio significativo. SIDH aveva poco più di un decennio di storia quando fu rotto, comparato ai quarant'anni di RSA o alle decadi di curve ellittiche classiche. La maturità di una congettura matematica, misurata dal tempo e dall'intensità dello scrutinio che ha ricevuto, è un indicatore cruciale della sua affidabilità. Questo non significa che nuove idee non debbano essere esplorate, ma che devono attraversare un lungo periodo di validazione prima di essere adottate per applicazioni critiche.

Terza lezione: la ridondanza nelle opzioni crittografiche è essenziale. Il fatto che il NIST avesse selezionato multipli finalisti basati su principi diversi (reticoli, codici, isogenie) si è rivelato cruciale. Il collasso di SIKE non ha compromesso l'intero progetto di standardizzazione post-quantistica perché altre alternative rimanevano disponibili.

Nonostante il fallimento di SIDH, la crittografia basata su isogenie non è morta. Varianti che non trasmettono punti ausiliari, come CSIDH (Commutative SIDH), rimangono apparentemente sicure, anche se con prestazioni inferiori. La ricerca continua su protocolli basati su isogenie che evitano le debolezze specifiche di SIDH, e la comunità ha acquisito una comprensione molto più profonda dei pericoli associati all'informazione ausiliaria nei protocolli basati su isogenie.

L'episodio SIDH serve anche come promemoria che il campo crittografico è dinamico e le congetture di durezza devono essere continuamente rivalutate alla luce di nuove tecniche di attacco. Un sistema considerato sicuro oggi potrebbe non esserlo domani, e la progettazione di infrastrutture crittografiche deve incorporare agilità crittografica—la capacità di migrare rapidamente ad alternative quando una primitiva diventa compromessa.

## **Parte VI: Sistemi Multivariati Quadratici e la Fragilità della Struttura Algebrica**

### **Il Problema MQ e i suoi Fondamenti Computazionali**

Il problema di risolvere sistemi di equazioni polinomiali multivariati su campi finiti è uno dei problemi computazionali più antichi e fondamentali in matematica. Nella sua forma generale, il problema MQ (Multivariate Quadratic) chiede di trovare soluzioni di un sistema di  $m$  equazioni quadratiche in  $n$  variabili su un campo finito  $F_q$ :

$$p_1(x_1, \dots, x_n) = 0 \quad p_2(x_1, \dots, x_n) = 0 \quad \vdots \quad p_m(x_1, \dots, x_n) = 0$$

dove ogni  $p_i$  è un polinomio quadratico:  $p_i(x) = \sum_{i,j} \alpha_{i,j} x_i x_j + \sum_i \beta_i x_i + \gamma_i$ .

Questo problema è stato dimostrato NP-completo da Garey e Johnson per campi finiti arbitrari. La dimostrazione riduce SAT (o altre varianti di problemi NP-completi) a MQ mediante una codifica algebrica appropriata. L'NP-completezza fornisce una giustificazione teorica forte per la durezza del problema nel caso peggiore: se  $P \neq NP$ , non esiste alcun algoritmo efficiente che risolva tutte le istanze di MQ.

La crittografia multivariata sfrutta questa durezza presumendo che, per sistemi appropriatamente scelti, il problema rimanga difficile anche in media. La congettura fondamentale è che sistemi casuali di equazioni quadratiche non abbiano struttura sfruttabile e richiedano tempo esponenziale per essere risolti, anche con algoritmi probabilistici o euristici.

Il campo della crittografia multivariata ha una storia travagliata, caratterizzata da numerosi schemi proposti e successivamente rotti. La tensione fondamentale è la seguente: per costruire uno schema crittografico utilizzabile, il possessore della chiave privata deve poter risolvere efficientemente il sistema MQ sottostante. Questo richiede che il sistema abbia una struttura nascosta che rende la soluzione facile per chi conosce il segreto. Ma questa stessa struttura può potenzialmente essere sfruttata da un attaccante per rompere il sistema. Il design di crittosistemi multivariati diventa quindi un esercizio delicato di mascheramento della struttura: abbastanza struttura per efficienza, mascherata sufficientemente bene da resistere agli attacchi.

## Algoritmi per Risolvere Sistemi MQ

Gli algoritmi per risolvere sistemi di equazioni polinomiali multivariati si dividono in approcci algebrici basati su basi di Gröbner e approcci combinatori di enumerazione esaustiva o ibridi.

Il metodo delle basi di Gröbner, sviluppato da Buchberger negli anni '60, fornisce un algoritmo generale per risolvere sistemi di equazioni polinomiali. L'idea è di trasformare il sistema originale in un sistema equivalente ma in "forma ridotta" da cui le soluzioni possono essere lette direttamente. Una base di Gröbner per un ideale  $I$  generato dai polinomi  $\{p_1, \dots, p_m\}$  è un insieme di generatori  $\{g_1, \dots, g_r\}$  con proprietà speciali che permettono la riduzione algoritmica.

L'algoritmo  $F_4$  di Faugère, e la sua variante più efficiente  $F_5$ , rappresentano lo stato dell'arte per il calcolo di basi di Gröbner. Questi algoritmi lavorano costruendo iterativamente una matrice contenente tutti i polinomi rilevanti fino a un certo grado, riducendola mediante algebra lineare, ed estraendo nuovi elementi della base di Gröbner dal risultato. Il processo termina quando non è possibile generare nuovi polinomi linearmente indipendenti.

La complessità del calcolo di basi di Gröbner dipende criticamente dal grado di regolarità  $d_{\text{reg}}$  del sistema. Per sistemi semi-regolari (una classe che include sistemi casuali con alta probabilità), il grado di regolarità può essere stimato usando la serie di Hilbert del sistema. Per un sistema di  $m$  equazioni quadratiche in  $n$  variabili su  $F_q$ , con  $m \geq n$ , il grado di regolarità è tipicamente  $d_{\text{reg}} \approx 2 + (n-1)/(m-1)$  per sistemi sovradeterminati ( $m > n$ ).

La complessità di  $F_4/F_5$  è dominata dall'algebra lineare sulla matrice di dimensione  $N \times N$  dove  $N$  è il numero di monomi di grado al più  $d_{\text{reg}}$ . Questo numero cresce come  $N \approx (n + d_{\text{reg}})$  sceglie  $d_{\text{reg}}$ , quindi per sistemi casuali con parametri bilanciati, la complessità è approssimativamente  $O(n^\omega \cdot d_{\text{reg}})$  dove  $\omega \approx 2.37$  è l'esponente dell'algebra lineare. Per  $d_{\text{reg}}$  costante o lentamente crescente, questo è polinomiale in  $n$ . Per  $d_{\text{reg}}$  che cresce linearmente con  $n$ , diventa esponenziale.

Nella pratica crittografica, i parametri sono scelti in modo che  $d_{\text{reg}}$  sia sufficientemente grande da rendere l'attacco computazionalmente infattibile. Per esempio, per sistemi con  $n = 256$  variabili e  $m = 256$  equazioni su  $F_{256}$ , il grado di regolarità atteso è circa  $d_{\text{reg}} \approx 4$ , e il calcolo della base di Gröbner richiede circa  $2^{70}$ - $2^{80}$  operazioni, approssimativamente equivalente a sicurezza a 70-80 bit.

Approcci ibridi combinano algebra lineare e enumerazione esaustiva. L'idea è di fissare (indovinare) i valori di  $k$  variabili, risolvere il sistema risultante di equazioni in  $n-k$  variabili usando basi di Gröbner, e ripetere per tutte le  $2^k$  scelte se si lavora su  $F_2$ . Bilanciando appropriatamente  $k$  con la complessità del risolutore per il sistema ridotto, si può ottenere una complessità totale inferiore agli approcci puri.

L'algoritmo XL (eXtended Linearization) e le sue varianti rappresentano un approccio diverso. XL moltiplica le equazioni originali per monomi di grado basso, generando un sistema sovradeterminato di equazioni di grado più alto. Quando il grado è sufficiente, il sistema può essere linearizzato trattando ogni monomio come una variabile indipendente, e risolto mediante algebra lineare. La complessità di XL è comparabile a  $F_4$  per sistemi semi-regolari, ma può essere più efficiente per sistemi con struttura particolare.

## Schemi di Firma Basati su MQ: Oil and Vinegar e Rainbow

Gli schemi di firma digitale basati su MQ hanno ricevuto particolare attenzione perché le firme digitali sono una primitiva crittografica essenziale e le alternative post-quantistiche sono limitate. Lo schema Oil and Vinegar, proposto da Patarin, e la sua generalizzazione Unbalanced Oil and Vinegar (UOV), hanno formato la base per numerose costruzioni successive.

La costruzione UOV divide le  $n$  variabili in due gruppi:  $v$  variabili "vinegar" e  $o$  variabili "oil", con  $v > o$ . La chiave privata consiste di un sistema di  $m$  equazioni quadratiche con una proprietà speciale: i termini quadratici coinvolgono solo prodotti di variabili vinegar con qualsiasi variabile (vinegar-vinegar e vinegar-oil), ma mai prodotti oil-oil. Formalmente:

$$p_i(x) = \sum_{\alpha, \beta \in V} \alpha_{i, \alpha, \beta} x_\alpha x_\beta + \sum_{\beta, \gamma \in O} \beta_{i, \beta, \gamma} x_\beta x_\gamma + \text{termini lineari e costante}$$

dove  $V$  è l'insieme degli indici vinegar e  $O$  l'insieme degli indici oil.

Questa struttura permette la firma efficiente come segue. Per firmare un messaggio hash  $h \in \mathbb{F}_q^m$ , il firmatario deve trovare una soluzione  $x$  di  $p(x) = h$ . Si scelgono valori casuali per le variabili vinegar  $x_V$ . Con questi valori fissati, il sistema diventa lineare nelle variabili oil  $x_O$ , e può essere risolto mediante algebra lineare in tempo  $O(o^3)$ . Se il sistema lineare risultante non ha soluzione (ciò accade con probabilità circa  $1/2$  se  $m = o$ ), si riprova con diversi valori vinegar.

La chiave pubblica è ottenuta "mascherando" la struttura oil-vinegar mediante trasformazioni affini. Specificamente, si applicano due trasformazioni invertibili  $S: \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$  e  $T: \mathbb{F}_q^m \rightarrow \mathbb{F}_q^m$ , e si pubblica il sistema  $P = T \circ p \circ S$ . Per un attaccante che vede solo  $P$ , la distinzione tra variabili oil e vinegar è nascosta, e  $P$  appare come un sistema casuale di equazioni quadratiche.

Lo schema Rainbow, proposto da Ding e Schmidt, generalizza UOV a una struttura a strati. Invece di una singola divisione oil-vinegar, Rainbow usa una sequenza di layer  $v_1 < v_2 < \dots < v_\ell < v_1 + o$  con equazioni che rispettano vincoli oil-vinegar layer-per-layer. Questo permette parametri più flessibili e firme più compatte.

Rainbow era uno dei finalisti del terzo round del processo di standardizzazione post-quantistica del NIST, considerato uno dei candidati più maturi per firme digitali post-quantistiche. I parametri suggeriti per sicurezza a 128 bit erano  $n = 96 + 72 = 168$  variabili con due layer, producendo firme di circa 64 byte e chiavi pubbliche di circa 150 KB.

## Il Collasso di Rainbow: L'Attacco di Beullens (2022)

Nel febbraio 2022, poche settimane dopo il collasso di SIDH, un altro finalista post-quantistico del NIST cadde sotto un attacco devastante. Ward Beullens presentò un attacco che recupera la chiave privata di Rainbow in tempo molto inferiore alle stime di sicurezza precedenti, riducendo la sicurezza effettiva dei parametri proposti da 128 bit a circa 40-60 bit, rendendo il sistema praticamente insicuro.

L'attacco di Beullens sfrutta la struttura a layer di Rainbow in modo sottile. L'intuizione chiave è che, anche se la trasformazione affine  $S$  mescola le variabili e nasconde la distinzione oil-vinegar, residui di questa struttura rimangono visibili attraverso proprietà algebriche del sistema pubblico  $P$ . Specificamente, Beullens mostrò che è possibile identificare sottospazi vettoriali correlati alla decomposizione oil-vinegar lavorando con intersezioni di kernel di certe mappe quadratiche derivate da  $P$ .

L'attacco procede in fasi. Prima, si costruisce una rappresentazione matriciale delle forme quadratiche associate a  $P$ . Per ogni equazione  $p_i$ , si associa una matrice simmetrica  $M_i$  tale che  $p_i(x) = x^T M_i x + \text{termini lineari}$ . Secondo, si cerca sottospazi invarianti sotto certe combinazioni lineari di queste matrici, sfruttando il fatto che la struttura a layer di Rainbow induce pattern algebrici riconoscibili.

Terzo, usando tecniche di riconciliazione e testing, si identificano progressivamente gli strati della decomposizione Rainbow. Ogni strato identificato riduce il problema a uno più piccolo, e il processo continua ricorsivamente fino a recuperare completamente la struttura oil-vinegar mascherata. L'attacco richiede principalmente algebra lineare—calcoli di kernel,

intersezioni di sottospazi, decomposizioni—ed evita completamente la necessità di risolvere sistemi MQ difficili.

La complessità dell'attacco per i parametri Rainbow standard è circa  $2^{45}$ – $2^{60}$  operazioni, molto inferiore ai  $2^{128}$  operazioni necessarie se il sistema fosse veramente indistinguibile da casuale. L'attacco è stato implementato e verificato sperimentalmente, confermando che Rainbow non fornisce il livello di sicurezza dichiarato.

Il collasso di Rainbow ha portato alla sua eliminazione dalla competizione NIST. Come nel caso di SIDH, la lezione è che struttura algebrica, anche quando attentamente mascherata, può lasciare "impronte digitali" sfruttabili. La crittografia multivariata rimane un'area attiva di ricerca, ma la sequenza di break—Miyaguchi, HFE, SFLASH, Rainbow—ha reso la comunità giustamente cauta.

Schemi alternativi come GeMSS (Great Multivariate Short Signature Scheme) e altri rimangono candidati, ma tutti portano il peso del sospetto che struttura nascosta possa essere eventualmente scoperta ed sfruttata. La tensione fondamentale della crittografia multivariata—bisogno di struttura per efficienza versus vulnerabilità attraverso quella struttura—rimane irrisolta.

## Parte VII: Funzioni Hash Crittografiche e la Congettura del Random Oracle

### Le Assunzioni Fondamentali sulle Funzioni Hash

Le funzioni hash crittografiche sono primitivi fondamentali che appaiono in quasi ogni protocollo crittografico moderno: firme digitali, schemi di commitment, proof of work per blockchain, derivazione di chiavi, costruzione di cifrari a blocchi, e molto altro. Una funzione hash crittografica è una funzione  $H: \{0,1\}^* \rightarrow \{0,1\}^n$  che mappa stringhe di lunghezza arbitraria a digest di lunghezza fissa  $n$  (tipicamente  $n = 256$  o  $512$ ).

Le proprietà di sicurezza richieste da una funzione hash crittografica sono tradizionalmente tre:

**Resistenza alla preimmagine:** Dato un digest  $h$ , deve essere computazionalmente difficile trovare un messaggio  $m$  tale che  $H(m) = h$ . Formalmente, per ogni algoritmo efficiente  $A$ , la probabilità che  $A$  riesca a trovare una preimmagine di un digest casuale deve essere trascurabile.

**Resistenza alla seconda preimmagine:** Dato un messaggio  $m_1$ , deve essere difficile trovare un messaggio diverso  $m_2 \neq m_1$  tale che  $H(m_1) = H(m_2)$ . Questo è più forte della resistenza alla preimmagine perché l'attaccante conosce già una preimmagine valida.

**Resistenza alle collisioni:** Deve essere difficile trovare qualsiasi coppia di messaggi distinti  $m_1, m_2$  con  $H(m_1) = H(m_2)$ . Questo è la proprietà più forte: l'attaccante ha completa libertà nella scelta di entrambi i messaggi.

Queste proprietà non sono dimostrazioni matematiche ma congetture empiriche. Non esiste alcuna dimostrazione che SHA-256, SHA-3, o qualsiasi altra funzione hash concreta soddisfi queste proprietà. L'evidenza della loro sicurezza si basa esclusivamente su decenni di tentativi falliti di trovare attacchi, combinati con principi di design che mirano a creare comportamento "caotico" e imprevedibile.

Il paradosso del compleanno fornisce un limite inferiore sulla difficoltà di trovare collisioni. Per una funzione hash con output di  $n$  bit, esistono  $2^n$  digest possibili. Se campioniamo casualmente circa  $2^{(n/2)}$  messaggi e computiamo i loro hash, ci aspettiamo di trovare una collisione con alta probabilità. Questo è dovuto al fatto che il numero di coppie distinte tra  $k$  messaggi è  $k(k-1)/2 \approx k^2/2$ , e quando  $k \approx 2^{(n/2)}$ , otteniamo circa  $2^n$  coppie, abbastanza per coprire lo spazio degli output con probabilità significativa.

Di conseguenza, per una funzione hash con output a  $n$  bit, la resistenza alle collisioni non può fornire più di  $n/2$  bit di sicurezza. Per SHA-256, questo significa al massimo 128 bit di sicurezza contro attacchi di collisione, anche se la funzione fosse perfettamente ideale. Per resistenza alla preimmagine, il limite è diverso: un attaccante deve testare approssimativamente  $2^n$  preimmagini prima di trovarne una valida, quindi la sicurezza teorica massima è  $n$  bit.

La congettura fondamentale sulle funzioni hash crittografiche moderne come SHA-256 o SHA-3 è che non esistano attacchi significativamente migliori degli attacchi generici basati sul paradosso del compleanno o sulla ricerca esaustiva. Per SHA-256, questo significa che trovare collisioni dovrebbe richiedere circa  $2^{128}$  operazioni, e trovare preimmagini dovrebbe richiedere circa  $2^{256}$  operazioni. Queste congetture non hanno dimostrazione matematica e si basano esclusivamente sulla fiducia nel design della funzione e sull'assenza di attacchi noti.

## Il Modello del Random Oracle: Astrazione e Realtà

Il modello del random oracle, introdotto formalmente da Bellare e Rogaway nel 1993, rappresenta uno strumento potente ma controverso nell'analisi di sicurezza crittografica. Un random oracle è una funzione matematica ideale  $H: \{0,1\}^* \rightarrow \{0,1\}^n$  con la seguente proprietà: per ogni input  $x$  mai interrogato prima,  $H(x)$  è scelto uniformemente a caso da  $\{0,1\}^n$  e indipendentemente da tutti i valori precedenti. Una volta che  $H(x)$  è stato definito, rimane consistente: successive interrogazioni con lo stesso input  $x$  ricevono la stessa risposta.

Nel modello del random oracle, tutti i partecipanti—inclusi gli attaccanti—hanno accesso a questo oracolo attraverso interrogazioni. L'oracolo è pubblico e deterministico (sempre la stessa risposta per lo stesso input), ma i suoi output sono perfettamente casuali e imprevedibili fino all'interrogazione.

La metodologia del random oracle per progettare sistemi crittografici è la seguente: primo, si progetta un protocollo assumendo l'esistenza di un random oracle. Si dimostra rigorosamente che il protocollo è sicuro nel modello del random oracle, tipicamente dimostrando che qualsiasi attacco riuscito potrebbe essere convertito in una soluzione a un problema computazionale difficile (come fattorizzazione o logaritmo discreto) con un numero

polinomiale di interrogazioni all'oracolo. Secondo, nella implementazione concreta, si sostituisce il random oracle con una funzione hash crittografica come SHA-256.

La congettura del random oracle afferma che questa sostituzione preserva la sicurezza: se un protocollo è sicuro nel modello del random oracle, allora l'istanziamento con una funzione hash "buona" produce un protocollo sicuro nel mondo reale. Questa congettura è estremamente utile metodologicamente perché permette di separare l'analisi della logica del protocollo dall'analisi della funzione hash specifica. Molti protocolli importanti, inclusi vari schemi di padding per RSA (come OAEP) e numerosi schemi di firma, hanno prove di sicurezza nel modello del random oracle.

Tuttavia, la congettura del random oracle è nota essere falsa in senso stretto. Canetti, Goldreich e Halevi costruirono nel 1998 uno schema artificiale che è dimostrabilmente sicuro nel modello del random oracle ma diventa insicuro per qualsiasi istanziamento concreto della funzione hash. La costruzione è altamente non naturale—incorpora esplicitamente nel protocollo una ricerca della descrizione della funzione hash per sfruttarla—ma dimostra matematicamente che la sostituzione random oracle  $\rightarrow$  funzione hash concreta non è sempre valida.

Nonostante questa separazione teorica, la comunità crittografica continua ad usare il modello del random oracle estensivamente per due ragioni. Prima, gli esempi di separazione noti sono tutti costruzioni artificiose che sembrano progettate specificamente per fallire, e nessun protocollo naturale è stato trovato sicuro nel random oracle model ma insicuro con funzioni hash concrete. Seconda, molti protocolli semplicemente non hanno prove di sicurezza alternative nel modello standard (senza random oracle), quindi il random oracle model rimane il miglior strumento disponibile per l'analisi.

La situazione rappresenta una tensione fondamentale nella crittografia moderna: il divario tra quello che possiamo dimostrare rigorosamente e quello che siamo disposti ad assumere nella pratica. Il modello del random oracle fornisce garanzie matematiche precise ma sotto assunzioni irrealistiche. L'alternativa—prove nel modello standard—è più desiderabile teoricamente ma spesso impossibile da ottenere o produce schemi con prestazioni inaccettabili.

## **Attacchi a Funzioni Hash: Storia di MD5 e SHA-1**

La storia degli attacchi a funzioni hash crittografiche fornisce lezioni importanti sulla fragilità delle costruzioni concrete e sull'importanza della conservatorismo nei parametri di sicurezza.

MD5, progettato da Ron Rivest nel 1991, era una funzione hash molto popolare negli anni '90 e primi 2000. Produce digest a 128 bit e era usato estensivamente per verificare l'integrità di file, in protocolli crittografici, e per hashing di password. La prima crepa nell'armatura di MD5 apparve nel 1996 quando Hans Dobbertin trovò collisioni nella funzione di compressione di MD5 (il componente interno), anche se non nell'intera funzione hash.

Nel 2004, Wang, Feng, Lai e Yu presentarono il primo attacco pratico di collisione contro MD5, riducendo la complessità da  $2^{64}$  (l'attacco del compleanno generico) a circa  $2^{39}$  operazioni. Questo fu un risultato drammatico: un attacco un milione di volte più veloce del

previsto. L'attacco sfrutta sottilmente la struttura algebrica della funzione di compressione di MD5, usando tecniche di differential cryptanalysis estese.

Successivi miglioramenti ridussero la complessità ulteriormente. Nel 2008, un attacco pratico fu dimostrato generando due certificati X.509 distinti con firme MD5 identiche, dimostrando che MD5 era completamente rotto per applicazioni di sicurezza. Oggi, collisioni MD5 possono essere generate in secondi su hardware moderno, e la funzione è considerata completamente insicura per qualsiasi applicazione crittografica.

SHA-1, progettato dalla NSA e standardizzato dal NIST nel 1995 come successore di MD5, produce digest a 160 bit. Per anni, SHA-1 fu considerato sicuro, con resistenza alle collisioni teorica di  $2^{80}$ . Nel 2005, Wang, Yin e Yu presentarono un attacco teorico che riduceva la complessità di collisione a circa  $2^{63}$  operazioni—significativamente migliore dell'attacco del compleanno ma non ancora pratico.

Nel 2017, il team di Google e CWI Amsterdam eseguì il primo attacco di collisione pratico contro SHA-1, chiamato "SHAttered". Generarono due documenti PDF distinti con lo stesso hash SHA-1, richiedendo circa  $2^{63}$  operazioni computazionali distribuite su molti mesi di calcolo. Questo dimostrava definitivamente che SHA-1 era rotto, portando alla sua deprecazione in applicazioni di sicurezza.

Gli attacchi a MD5 e SHA-1 condividono tecniche comuni. Entrambi usano differential cryptanalysis, dove l'attaccante costruisce messaggi con differenze specifiche progettate per propagare attraverso la funzione in modo controllato. L'obiettivo è trovare due messaggi  $m_1$ ,  $m_2$  tali che le differenze introdotte nei vari round della funzione si cancellino alla fine, producendo lo stesso digest.

La struttura Merkle-Damgård, usata sia da MD5 che SHA-1, ha anche una debolezza intrinseca: se si può generare una collisione per messaggi di una certa lunghezza, quella collisione può essere estesa a messaggi più lunghi semplicemente appendendo gli stessi dati a entrambi i messaggi collidenti. Questa proprietà di estensione della collisione rende gli attacchi più potenti di quanto potrebbero apparire inizialmente.

## **SHA-2, SHA-3 e lo Stato Attuale delle Funzioni Hash**

Dopo i problemi con MD5 e SHA-1, la comunità crittografica si è mossa verso la famiglia SHA-2, che include SHA-224, SHA-256, SHA-384 e SHA-512. Questi sono varianti con diverse lunghezze di output ma struttura interna simile, basata ancora sul paradigma Merkle-Damgård ma con una funzione di compressione completamente riprogettata.

SHA-256, il membro più popolare della famiglia, produce digest a 256 bit e dovrebbe fornire 128 bit di sicurezza contro collisioni e 256 bit contro attacchi di preimmagine. Nonostante quasi due decenni di analisi intensiva, nessun attacco significativo contro SHA-256 è stato trovato. Gli attacchi migliori conosciuti funzionano su versioni ridotte della funzione (meno round), ma non sulla versione completa. Questo record di resistenza ha reso SHA-256 la funzione hash standard de facto per applicazioni moderne, inclusa Bitcoin e altre blockchain.

In parallelo, il NIST condusse una competizione dal 2007 al 2012 per progettare una nuova funzione hash standard che fosse strutturalmente diversa da SHA-2, fornendo diversità e

ridondanza nel caso SHA-2 fosse compromessa. Il vincitore fu Keccak, standardizzato come SHA-3.

SHA-3 usa una costruzione completamente diversa chiamata sponge construction. Invece di Merkle-Damgård, la sponge function mantiene uno stato interno di  $b$  bit diviso in due parti: una "rate"  $r$  e una "capacity"  $c$  con  $r + c = b$ . I dati di input sono "assorbiti" nello stato mediante XOR con la porzione rate, seguita dall'applicazione di una permutazione. Dopo aver assorbito tutto l'input, la funzione "spreme" l'output estraendo bit dalla porzione rate e applicando la permutazione ripetutamente.

La permutazione centrale di SHA-3, chiamata Keccak-f, opera su uno stato tridimensionale di  $5 \times 5$  lane di 64 bit ciascuna, per un totale di 1600 bit. La permutazione consiste di 24 round, ciascuno applicando una sequenza di trasformazioni non lineari che mescolano lo stato in modo complesso. Il design è elegante e simmetrico, ispirato da principi di crittografia e teoria dell'informazione.

SHA-3 ha il vantaggio di essere completamente diverso da SHA-2, quindi un attacco che rompe SHA-2 probabilmente non si trasferisce a SHA-3. Questo fornisce robustezza attraverso diversità. Tuttavia, SHA-3 è computazionalmente più costoso di SHA-256 in software, e SHA-256 ha beneficiato di più tempo per analisi e ottimizzazione hardware. Di conseguenza, nella pratica, SHA-256 rimane più comunemente usato, con SHA-3 disponibile come alternativa per applicazioni che richiedono assicurazione extra o hanno requisiti specifici che SHA-3 soddisfa meglio.

## Funzioni Hash e Blockchain: Requisiti Speciali

L'avvento di Bitcoin e blockchain ha introdotto un nuovo uso delle funzioni hash: proof of work. In Bitcoin, i miner competono per trovare un nonce tale che il hash SHA-256 del blocco (che include il nonce) soddisfi una certa condizione di difficoltà—tipicamente che il digest abbia un certo numero di zeri leading.

Questo uso richiede proprietà leggermente diverse dalle applicazioni crittografiche tradizionali. Specificamente, la funzione hash deve essere difficile da accelerare mediante hardware specializzato (ASIC). Se gli ASIC possono computare hash migliaia di volte più efficientemente di GPU o CPU generali, la rete diventa centralizzata in coloro che possono permettersi hardware costoso.

Alcune blockchain hanno adottato funzioni hash "ASIC-resistant" progettate per essere difficili da implementare efficientemente in hardware custom. Scrypt, usato da Litecoin, incorpora operazioni memory-hard che richiedono accesso a grandi quantità di memoria, rendendo gli ASIC meno vantaggiosi. Ethash, usato originariamente da Ethereum, usa un DAG (Directed Acyclic Graph) molto grande che deve essere memorizzato per mining efficiente.

Tuttavia, la resistenza ASIC si è dimostrata effimera. Per ogni funzione "ASIC-resistant" proposta, ingegneri hardware motivati hanno eventualmente progettato ASIC efficienti, spesso con vantaggi di 10-100x su hardware generale. Questo ha portato alcuni a concludere che la resistenza ASIC è un obiettivo impossibile: qualsiasi computazione

deterministica può essere ottimizzata in hardware custom se l'incentivo economico è sufficiente.

## Parte VIII: Il Problema del Sottogruppo Nascosto e Minacce Quantistiche Future

### Fondamenti della Computazione Quantistica e Algoritmi di Ricerca

Per comprendere appieno le minacce future alla crittografia, dobbiamo esaminare le capacità dei computer quantistici oltre gli algoritmi di Shor e Grover che già conosciamo. Il problema del sottogruppo nascosto (Hidden Subgroup Problem, HSP) fornisce un framework unificato per comprendere molti algoritmi quantistici, incluso Shor come caso speciale.

Il problema del sottogruppo nascosto è formulato come segue. Sia  $G$  un gruppo finito e sia  $f: G \rightarrow S$  una funzione da  $G$  a un insieme finito  $S$ . Supponiamo che esista un sottogruppo  $H$  di  $G$  tale che  $f$  sia costante e distinta su ogni coset di  $H$ , cioè  $f(x) = f(y)$  se e solo se  $x$  e  $y$  appartengono allo stesso coset  $gH$  per qualche  $g \in G$ . Il problema HSP chiede di determinare il sottogruppo  $H$  dato accesso a interrogazioni di  $f$ .

Per gruppi abeliani, esiste un algoritmo quantistico efficiente per HSP basato sulla quantum Fourier transform (QFT). Questo algoritmo è precisamente quello che sottende l'algoritmo di Shor per la fattorizzazione e il logaritmo discreto. Per la fattorizzazione di  $n = pq$ , il gruppo è  $G = \mathbb{Z}$  con  $f(x) = a^x \bmod n$ , e il sottogruppo nascosto è  $H = r\mathbb{Z}$  dove  $r$  è l'ordine di  $a$  modulo  $n$ . La QFT estrae  $r$  efficientemente, e da  $r$  si ottiene la fattorizzazione.

Per gruppi non abeliani, la situazione è molto più complessa e costituisce una delle frontiere principali della ricerca in algoritmica quantistica. Se esistesse un algoritmo quantistico efficiente per HSP su gruppi non abeliani arbitrari, numerosi problemi crittografici aggiuntivi collasserebbero. In particolare, molti schemi crittografici basati su gruppi non abeliani, alcuni dei quali sono stati proposti come candidati post-quantistici, sarebbero vulnerabili.

Il problema del grafo di isomorfismo è un esempio importante correlato a HSP su gruppi non abeliani. Dato un gruppo di permutazioni  $G$  e una funzione  $f$  che rappresenta l'azione del gruppo su un grafo, determinare se due grafi sono isomorfi si riduce a HSP sul gruppo simmetrico  $S_n$ . Nonostante decenni di ricerca, non è noto se HSP su  $S_n$  possa essere risolto efficientemente quantisticamente, e di conseguenza lo status dell'isomorfismo di grafi contro computer quantistici rimane aperto.

La situazione per gruppi diedrali—un caso speciale di gruppi non abeliani—è più promettente per gli attaccanti. Regev mostrò che HSP sul gruppo diedrale  $D_n$  può essere risolto quantisticamente con complessità polinomiale, anche se l'algoritmo è tecnicamente più complesso di quello per gruppi abeliani. Questo risultato ha implicazioni per alcuni schemi crittografici basati su reticoli, anche se non per i candidati principali come LWE e NTRU.

### Crittografia Quantistica: Distribuzione Quantistica di Chiavi

Mentre i computer quantistici minacciano la crittografia classica, la meccanica quantistica offre anche nuove possibilità per la sicurezza crittografica. La distribuzione quantistica di chiavi (Quantum Key Distribution, QKD) permette a due parti di stabilire una chiave segreta condivisa con sicurezza garantita dalle leggi della fisica quantistica, indipendentemente dalla potenza computazionale dell'attaccante.

Il protocollo BB84, proposto da Bennett e Brassard nel 1984, illustra i principi fondamentali. Alice prepara una sequenza di qubit, ciascuno in uno di quattro stati possibili:  $|0\rangle$ ,  $|1\rangle$ ,  $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ , o  $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ . Gli stati  $|0\rangle$ ,  $|1\rangle$  appartengono alla base computazionale, mentre  $|+\rangle$ ,  $|-\rangle$  appartengono alla base di Hadamard. Alice sceglie casualmente quale base usare per ogni qubit e quale dei due stati in quella base preparare.

Alice invia i qubit a Bob attraverso un canale quantistico. Bob riceve ciascun qubit e sceglie casualmente una base (computazionale o Hadamard) nella quale misurarlo. Se Bob sceglie la stessa base che Alice usò per preparare il qubit, la sua misurazione rivelerà il bit corretto con certezza. Se sceglie la base sbagliata, ottiene un risultato casuale.

Dopo la trasmissione, Alice e Bob comunicano pubblicamente quali basi usarono per ciascun qubit (ma non i valori dei bit). Scartano tutti i qubit dove usarono basi diverse, mantenendo solo quelli dove le basi coincidevano. Questi bit concordati formano la chiave grezza.

La sicurezza deriva dal principio di non-clonazione quantistica: un attaccante Eve che intercetta i qubit non può copiarli perfettamente o misurarli senza disturbare il loro stato. Se Eve misura un qubit, introduce errori che Alice e Bob possono rilevare confrontando un sottoinsieme casuale dei loro bit. Un tasso di errore superiore alla soglia attesa dal rumore naturale indica la presenza di intercettazione, e Alice e Bob abortiscono il protocollo.

Dopo aver verificato che il tasso di errore è accettabile, Alice e Bob applicano information reconciliation per correggere errori rimanenti e privacy amplification per rimuovere qualsiasi informazione parziale che Eve potrebbe aver acquisito. Il risultato finale è una chiave segreta condivisa con sicurezza dimostrabile basata sulla meccanica quantistica.

QKD è stato implementato con successo in fibra ottica su distanze di centinaia di chilometri e tramite free-space optical links a satelliti. Tuttavia, QKD ha limitazioni pratiche significative. Richiede canali quantistici specializzati—non può essere implementato su Internet esistente. Le distanze sono limitate dalla perdita nei canali ottici, anche se i ripetitori quantistici (ancora tecnologia in sviluppo) potrebbero eventualmente estendere il range. Inoltre, QKD stabilisce solo chiavi simmetriche, non sostituisce completamente la crittografia a chiave pubblica per funzionalità come firme digitali.

La sicurezza di QKD è anche soggetta a considerazioni pratiche. Mentre la teoria garantisce sicurezza perfetta, le implementazioni reali hanno vulnerabilità attraverso side channels e imperfezioni hardware. Numerosi attacchi contro implementazioni QKD commerciali sono stati dimostrati, sfruttando debolezze come rilevatori imperfetti, timing attacks, o trojan horses ottici. Questi attacchi non violano i principi quantistici fondamentali ma dimostrano il divario tra sicurezza teorica e pratica.

## **Crittografia Post-Quantistica: Lo Stato della Standardizzazione**

Il processo di standardizzazione post-quantistica del NIST, iniziato nel 2016, rappresenta uno degli sforzi più significativi nella storia moderna della crittografia per preparare la transizione all'era quantistica. Dopo multipli round di valutazione, nel luglio 2022 il NIST annunciò i primi algoritmi selezionati per standardizzazione.

Per cifratura e key encapsulation, il vincitore principale è CRYSTALS-Kyber, un sistema basato su Module-LWE. Kyber offre un buon equilibrio tra sicurezza, prestazioni e dimensioni di chiavi/testo cifrato. I parametri Kyber-512 mirano a sicurezza quantistica equivalente a AES-128, mentre Kyber-1024 mira all'equivalente di AES-256. Le chiavi pubbliche sono circa 800-1600 byte, e i testi cifrati circa 700-1500 byte, dipendendo dal livello di sicurezza.

Per firme digitali, tre schemi furono selezionati: CRYSTALS-Dilithium (basato su Module-LWE), FALCON (basato su NTRU lattices), e SPHINCS+ (basato su funzioni hash). Dilithium produce firme di 2-4 KB con chiavi pubbliche di 1-2 KB. FALCON offre firme più compatte (circa 700 byte) ma richiede generazione di chiavi più complessa. SPHINCS+ è completamente basato su hash, fornendo massima conservatorismo contro attacchi sia classici che quantistici, ma con firme molto più grandi (8-50 KB).

Dopo il collasso di SIKE, il NIST avviò un quarto round per identificare algoritmi alternativi basati su principi diversi dai reticoli, per diversificare il portfolio di standard. I finalisti includono Classic McEliece (basato su codici), BIKE e HQC (anche su codici), e SIKE fu escluso. La diversificazione è cruciale: se una debolezza fondamentale nei problemi su reticoli fosse scoperta, l'intera infrastruttura post-quantistica basata su LWE collasserebbe senza alternative mature.

La transizione pratica alla crittografia post-quantistica presenta sfide enormi. I protocolli esistenti—TLS, IPsec, SSH, VPN—sono profondamente integrati con crittografia classica. Migrare richiede aggiornamenti software su miliardi di dispositivi, molti dei quali sono sistemi embedded o legacy difficili da aggiornare. Le dimensioni maggiori delle chiavi e testi cifrati post-quantistici possono causare problemi di compatibilità con protocolli che assumono limiti sulle dimensioni dei messaggi.

La strategia di hybrid encryption è stata proposta per la transizione: combinare algoritmi classici e post-quantistici, richiedendo che entrambi siano rotti per compromettere la sicurezza. Ad esempio, TLS 1.3 potrebbe usare sia ECDHE (Elliptic Curve Diffie-Hellman Ephemeral) che Kyber simultaneamente, derivando la chiave di sessione da entrambi. Questo fornisce sicurezza contro attaccanti classici anche se Kyber fosse debole, e sicurezza quantistica anche se ECDH fosse compromesso da computer quantistici.

## **Timing delle Minacce Quantistiche: "Store Now, Decrypt Later"**

Una considerazione critica nella transizione post-quantistica è l'attacco "store now, decrypt later" (SNDL). Un attaccante può intercettare e memorizzare comunicazioni cifrate oggi, anche senza la capacità di decifrarle. Quando computer quantistici sufficientemente potenti diventano disponibili—potenzialmente tra 10-30 anni—l'attaccante può retroattivamente decifrare i dati memorizzati.

Questo scenario è particolarmente preoccupante per dati con valore a lungo termine: segreti di stato, proprietà intellettuale, informazioni mediche, dati biometrici. Anche se un'organizzazione pianifica di migrare a crittografia post-quantistica entro dieci anni, i dati trasmessi oggi potrebbero già essere compromessi se un attaccante li sta memorizzando.

Le agenzie di sicurezza nazionali, incluse NSA e ANSSI (l'agenzia francese), raccomandano di iniziare la transizione immediatamente, specialmente per dati classificati o con valore a lungo termine. La NSA pubblicò nel 2022 una Commercial National Security Algorithm (CNSA) Suite 2.0 che include algoritmi post-quantistici e stabilisce timeline per la migrazione.

La finestra temporale per la transizione è stretta. Sviluppare, testare e deployare nuovi standard crittografici richiede anni o decenni. Sistemi deployed oggi potrebbero rimanere in uso per 20-30 anni. Se computer quantistici capaci di rompere RSA-2048 emergono nel 2040, sistemi deployed nel 2025 con vita di 20 anni sarebbero vulnerabili per metà della loro vita operativa.

Previsioni su quando computer quantistici crittograficamente rilevanti emergeranno variano ampiamente. Ottimisti suggeriscono 10-15 anni, pessimisti oltre 50 anni, e alcuni esperti dubitano che scalare a milioni di qubit logici necessari sia fattibile con tecnologie note. Indipendentemente dalla timeline esatta, il principio precauzionale suggerisce di prepararsi ora piuttosto che aspettare la certezza.

## Conclusione: Vivere con l'Incertezza Matematica

La crittografia moderna opera in uno spazio concettuale peculiare: proteggere comunicazioni e dati del valore di trilioni di dollari utilizzando sistemi la cui sicurezza non può essere dimostrata matematicamente. Ogni transazione con carta di credito, ogni connessione HTTPS, ogni firma digitale su contratti legali si basa su congetture—assunzioni matematiche che crediamo vere ma non possiamo provare.

Questo stato di affari potrebbe sembrare precario, persino irresponsabile. Come possiamo affidare la sicurezza della società moderna a fondazioni così incerte? La risposta risiede nel riconoscere che l'incertezza matematica non implica debolezza pratica. Le congetture su cui si basa la crittografia—durezza della fattorizzazione, del logaritmo discreto, di SVP, di LWE—hanno resistito a decenni di scrutinio intensivo da parte di comunità di ricercatori dedicati e ben finanziati. Gli algoritmi più potenti noti rimangono esponenziali, e continui miglioramenti nelle tecniche di attacco sono stati controbilanciati da aumenti nelle dimensioni dei parametri.

Tuttavia, la storia recente—il collasso di SIDH nel 2022, di Rainbow nello stesso anno, di SIKE poco dopo—serve come promemoria che le congetture matematiche sono fragili. Un singolo insight può distruggere anni di fiducia consolidata. La ricerca crittografica deve quindi procedere con umiltà epistemologica, riconoscendo che oggi teniamo per vere assunzioni che potrebbero rivelarsi false domani.

La gestione del rischio in crittografia richiede diversificazione. Non dovremmo affidarci a una singola congettura o famiglia di problemi computazionali. Gli standard moderni incorporano

multipli schemi basati su principi diversi: reticoli, codici, hash, isogenie. Se una famiglia collassa, altre rimangono. L'ibridazione—combinare schemi classici e post-quantistici—fornisce difesa in profondità contro minacce sia presenti che future.

La transizione verso la crittografia post-quantistica rappresenta una delle sfide tecnologiche più significative che la società affronterà nei prossimi due decenni. A differenza delle transizioni precedenti, come il passaggio da DES ad AES o l'adozione di curve ellittiche, questa transizione è motivata non da debolezze scoperte ma da minacce anticipate. Dobbiamo ricostruire l'infrastruttura crittografica globale in preparazione per computer quantistici che non esistono ancora e potrebbero non esistere per decenni, se mai.

Questa natura preventiva rende la transizione politicamente e economicamente difficile. Le organizzazioni resistono al costo di aggiornare sistemi funzionanti per mitigare minacce speculative. Tuttavia, la natura dell'attacco "store now, decrypt later" elimina la possibilità di attendere la certezza prima di agire. Quando computer quantistici crittograficamente rilevanti emergeranno, sarà troppo tardi per proteggere i dati già trasmessi.

La crittografia rimane, alla fine, un'arte tanto quanto una scienza. Combina rigore matematico con intuizione euristica, prove formali con evidenza empirica, principi teorici con considerazioni ingegneristiche pratiche. Le congetture che sottendono la crittografia non sono verità matematiche assolute ma piuttosto il miglior compromesso che abbiamo trovato tra sicurezza dimostrabile e efficienza pratica.

Mentre continuiamo a costruire un mondo sempre più digitale e interconnesso, la nostra dipendenza da queste congetture non dimostrate crescerà. L'infrastruttura critica, i servizi finanziari, i sistemi sanitari, le comunicazioni governative—tutti si affidano alla validità continuata di assunzioni matematiche che nessuno ha dimostrato e che potrebbero, in linea di principio, essere false. Questa è una vulnerabilità sistemica che nessuna quantità di ingegneria può eliminare completamente.

Ciò che possiamo fare è continuare il lavoro: scrutinare le congetture esistenti con sempre maggior rigore, ricercare nuove primitivi crittografiche con fondazioni teoriche più solide, mantenere diversità negli approcci per evitare single points of failure, e prepararci per transizioni future quando congetture attuali potrebbero cadere. La crittografia è una corsa continua tra progettisti e attaccanti, e la nostra sicurezza collettiva dipende dal rimanere sempre un passo avanti.

In questo senso, le congetture fondamentali della crittografia moderna non sono solo problemi matematici astratti ma questioni di sicurezza nazionale e globale. La loro validità determina se le nostre comunicazioni rimangono private, se le nostre transazioni finanziarie sono sicure, se i nostri segreti rimangono segreti. Viviamo nell'era delle congetture, e la nostra società è costruita su fondamenta che crediamo solide ma non possiamo provare inattaccabili. È una situazione scomoda, ma per ora, è la migliore che abbiamo.